

	1010101000_1010000101	// W0752_2048 = -0.671559	-0.740951
	1010100111_1010000110	// W0753_2048 = -0.673829	-0.738887
	1010100110_1010000111	// W0754_2048 = -0.676093	-0.736817
	1010100100_1010001001	// W0756_2048 = -0.680601	-0.732654
5	1010100001_1010001011	// W0758_2048 = -0.685084	-0.728464
	1010100000_1010001100	// W0759_2048 = -0.687315	-0.726359
	1010011111_1010001101	// W0760_2048 = -0.689541	-0.724247
	1010011101_1010001111	// W0762_2048 = -0.693971	-0.720003
	1010011010_1010010010	// W0764_2048 = -0.698376	-0.715731
10	1010011001_1010010011	// W0765_2048 = -0.700569	-0.713585
	1010011000_1010010100	// W0766_2048 = -0.702755	-0.711432
	1010010110_1010010110	// W0768_2048 = -0.707107	-0.707107
	1010010100_1010011000	// W0770_2048 = -0.711432	-0.702755
	1010010011_1010011001	// W0771_2048 = -0.713585	-0.700569
15	1010010010_1010011010	// W0772_2048 = -0.715731	-0.698376
	1010001111_1010011101	// W0774_2048 = -0.720003	-0.693971
	1010001101_1010011111	// W0776_2048 = -0.724247	-0.689541
	1010001100_1010100000	// W0777_2048 = -0.726359	-0.687315
	1010001011_1010100001	// W0778_2048 = -0.728464	-0.685084
20	1010001001_1010100100	// W0780_2048 = -0.732654	-0.680601
	1010000111_1010100110	// W0782_2048 = -0.736817	-0.676093
	1010000110_1010100111	// W0783_2048 = -0.738887	-0.673829
	1010000101_1010101000	// W0784_2048 = -0.740951	-0.671559
	1010000011_1010101010	// W0786_2048 = -0.745058	-0.667000
25	1010000000_1010101101	// W0788_2048 = -0.749136	-0.662416
	1001111111_1010101110	// W0789_2048 = -0.751165	-0.660114
	1001111110_1010101111	// W0790_2048 = -0.753187	-0.657807
	1001111100_1010110010	// W0792_2048 = -0.757209	-0.653173
	1001111010_1010110100	// W0794_2048 = -0.761202	-0.648514
30	1001111001_1010110101	// W0795_2048 = -0.763188	-0.646176
	1001111000_1010110110	// W0796_2048 = -0.765167	-0.643832
	1001110110_1010111001	// W0798_2048 = -0.769103	-0.639124
	1001110100_1010111011	// W0800_2048 = -0.773010	-0.634393
	1001110011_1010111100	// W0801_2048 = -0.774953	-0.632019
35	1001110010_1010111110	// W0802_2048 = -0.776888	-0.629638
	1001110000_1011000000	// W0804_2048 = -0.780737	-0.624859
	1001101110_1011000011	// W0806_2048 = -0.784557	-0.620057
	1001101101_1011000100	// W0807_2048 = -0.786455	-0.617647
	1001101100_1011000101	// W0808_2048 = -0.788346	-0.615232
40	1001101010_1011000111	// W0810_2048 = -0.792107	-0.610383
	1001101001_1011001010	// W0812_2048 = -0.795837	-0.605511
	1001101000_1011001011	// W0813_2048 = -0.797691	-0.603067
	1001100111_1011001100	// W0814_2048 = -0.799537	-0.600616
	1001100101_1011001111	// W0816_2048 = -0.803208	-0.595699
45	1001100011_1011010010	// W0818_2048 = -0.806848	-0.590760
	1001100010_1011010011	// W0819_2048 = -0.808656	-0.588282
	1001100001_1011010100	// W0820_2048 = -0.810457	-0.585798
	1001011111_1011010111	// W0822_2048 = -0.814036	-0.580814
	1001011101_1011011001	// W0824_2048 = -0.817585	-0.575808
50	1001011100_1011011010	// W0825_2048 = -0.819348	-0.573297
	1001011100_1011011100	// W0826_2048 = -0.821103	-0.570781
	1001011010_1011011110	// W0828_2048 = -0.824589	-0.565732
	1001011000_1011100001	// W0830_2048 = -0.828045	-0.560662
	1001010111_1011100010	// W0831_2048 = -0.829761	-0.558119
55	1001010110_1011100100	// W0832_2048 = -0.831470	-0.555570
	1001010101_1011100110	// W0834_2048 = -0.834863	-0.550458

	1001010011_1011101001	// W0836_2048 = -0.838225	-0.545325
	1001010010_1011101010	// W0837_2048 = -0.839894	-0.542751
	1001010001_1011101011	// W0838_2048 = -0.841555	-0.540171
	1001001111_1011101110	// W0840_2048 = -0.844854	-0.534998
5	1001001110_1011110001	// W0842_2048 = -0.848120	-0.529804
	1001001101_1011110010	// W0843_2048 = -0.849742	-0.527199
	1001001100_1011110011	// W0844_2048 = -0.851355	-0.524590
	1001001010_1011110110	// W0846_2048 = -0.854558	-0.519356
	1001001001_1011111001	// W0848_2048 = -0.857729	-0.514103
10	1001001000_1011111010	// W0849_2048 = -0.859302	-0.511469
	1001000111_1011111011	// W0850_2048 = -0.860867	-0.508830
	1001000110_1011111110	// W0852_2048 = -0.863973	-0.503538
	1001000100_1100000001	// W0854_2048 = -0.867046	-0.498228
	1001000011_1100000010	// W0855_2048 = -0.868571	-0.495565
15	1001000011_1100000100	// W0856_2048 = -0.870087	-0.492898
	1001000001_1100000110	// W0858_2048 = -0.873095	-0.487550
	1000111111_1100001001	// W0860_2048 = -0.876070	-0.482184
	1000111111_1100001010	// W0861_2048 = -0.877545	-0.479494
	1000111110_1100001100	// W0862_2048 = -0.879012	-0.476799
20	1000111100_1100001111	// W0864_2048 = -0.881921	-0.471397
	1000111011_1100010001	// W0866_2048 = -0.884797	-0.465976
	1000111010_1100010011	// W0867_2048 = -0.886223	-0.463260
	1000111010_1100010100	// W0868_2048 = -0.887640	-0.460539
	1000111000_1100010111	// W0870_2048 = -0.890449	-0.455084
25	1000110111_1100011010	// W0872_2048 = -0.893224	-0.449611
	1000110110_1100011011	// W0873_2048 = -0.894599	-0.446869
	1000110101_1100011101	// W0874_2048 = -0.895966	-0.444122
	1000110100_1100011111	// W0876_2048 = -0.898674	-0.438616
	1000110011_1100100010	// W0878_2048 = -0.901349	-0.433094
30	1000110010_1100100100	// W0879_2048 = -0.902673	-0.430326
	1000110001_1100100101	// W0880_2048 = -0.903989	-0.427555
	1000110000_1100101000	// W0882_2048 = -0.906596	-0.422000
	1000101111_1100101011	// W0884_2048 = -0.909168	-0.416430
	1000101110_1100101100	// W0885_2048 = -0.910441	-0.413638
35	1000101101_1100101110	// W0886_2048 = -0.911706	-0.410843
	1000101100_1100110001	// W0888_2048 = -0.914210	-0.405241
	1000101011_1100110011	// W0890_2048 = -0.916679	-0.399624
	1000101010_1100110101	// W0891_2048 = -0.917901	-0.396810
	1000101001_1100110110	// W0892_2048 = -0.919114	-0.393992
40	1000101000_1100111001	// W0894_2048 = -0.921514	-0.388345
	1000100111_1100111100	// W0896_2048 = -0.923880	-0.382683
	1000100110_1100111110	// W0897_2048 = -0.925049	-0.379847
	1000100110_1100111111	// W0898_2048 = -0.926210	-0.377007
	1000100101_1101000010	// W0900_2048 = -0.928506	-0.371317
45	1000100011_1101000101	// W0902_2048 = -0.930767	-0.365613
	1000100011_1101000110	// W0903_2048 = -0.931884	-0.362756
	1000100010_1101001000	// W0904_2048 = -0.932993	-0.359895
	1000100001_1101001011	// W0906_2048 = -0.935184	-0.354164
	1000100000_1101001110	// W0908_2048 = -0.937339	-0.348419
50	1000100000_1101001111	// W0909_2048 = -0.938404	-0.345541
	1000011111_1101010001	// W0910_2048 = -0.939459	-0.342661
	1000011110_1101010100	// W0912_2048 = -0.941544	-0.336890
	1000011101_1101010110	// W0914_2048 = -0.943593	-0.331106
	1000011100_1101011000	// W0915_2048 = -0.944605	-0.328210
55	1000011100_1101011001	// W0916_2048 = -0.945607	-0.325310
	1000011011_1101011100	// W0918_2048 = -0.947586	-0.319502

	1000011010_1101011111	// W0920_2048 = -0.949528	-0.313682
	1000011001_1101100001	// W0921_2048 = -0.950486	-0.310767
	1000011001_1101100010	// W0922_2048 = -0.951435	-0.307850
	1000011000_1101100101	// W0924_2048 = -0.953306	-0.302006
5	1000010111_1101101000	// W0926_2048 = -0.955141	-0.296151
	1000010111_1101101010	// W0927_2048 = -0.956045	-0.293219
	1000010110_1101101011	// W0928_2048 = -0.956940	-0.290285
	1000010101_1101101110	// W0930_2048 = -0.958703	-0.284408
	1000010100_1101110001	// W0932_2048 = -0.960431	-0.278520
10	1000010100_1101110011	// W0933_2048 = -0.961280	-0.275572
	1000010011_1101110100	// W0934_2048 = -0.962121	-0.272621
	1000010011_1101110111	// W0936_2048 = -0.963776	-0.266713
	1000010010_1101111010	// W0938_2048 = -0.965394	-0.260794
	1000010001_1101111100	// W0939_2048 = -0.966190	-0.257831
15	1000010001_1101111110	// W0940_2048 = -0.966976	-0.254866
	1000010000_1110000001	// W0942_2048 = -0.968522	-0.248928
	1000001111_1110000100	// W0944_2048 = -0.970031	-0.242980
	1000001111_1110000101	// W0945_2048 = -0.970772	-0.240003
	1000001111_1110000111	// W0946_2048 = -0.971504	-0.237024
20	1000001110_1110001010	// W0948_2048 = -0.972940	-0.231058
	1000001101_1110001101	// W0950_2048 = -0.974339	-0.225084
	1000001101_1110001110	// W0951_2048 = -0.975025	-0.222094
	1000001100_1110010000	// W0952_2048 = -0.975702	-0.219101
	1000001100_1110010011	// W0954_2048 = -0.977028	-0.213110
25	1000001011_1110010110	// W0956_2048 = -0.978317	-0.207111
	1000001011_1110010111	// W0957_2048 = -0.978948	-0.204109
	1000001010_1110011001	// W0958_2048 = -0.979570	-0.201105
	1000001010_1110011100	// W0960_2048 = -0.980785	-0.195090
	1000001001_1110011111	// W0962_2048 = -0.981964	-0.189069
30	1000001001_1110100001	// W0963_2048 = -0.982539	-0.186055
	1000001001_1110100010	// W0964_2048 = -0.983105	-0.183040
	1000001000_1110100101	// W0966_2048 = -0.984210	-0.177004
	1000001000_1110101000	// W0968_2048 = -0.985278	-0.170962
	1000000111_1110101010	// W0969_2048 = -0.985798	-0.167938
35	1000000111_1110101100	// W0970_2048 = -0.986308	-0.164913
	1000000111_1110101111	// W0972_2048 = -0.987301	-0.158858
	1000000110_1110110010	// W0974_2048 = -0.988258	-0.152797
	1000000110_1110110011	// W0975_2048 = -0.988722	-0.149765
	1000000110_1110110101	// W0976_2048 = -0.989177	-0.146730
40	1000000101_1110111000	// W0978_2048 = -0.990058	-0.140658
	1000000101_1110111011	// W0980_2048 = -0.990903	-0.134581
	1000000100_1110111101	// W0981_2048 = -0.991311	-0.131540
	1000000100_1110111110	// W0982_2048 = -0.991710	-0.128498
	1000000100_1111000001	// W0984_2048 = -0.992480	-0.122411
45	1000000011_1111000100	// W0986_2048 = -0.993212	-0.116319
	1000000011_1111000110	// W0987_2048 = -0.993564	-0.113271
	1000000011_1111001000	// W0988_2048 = -0.993907	-0.110222
	1000000011_1111001011	// W0990_2048 = -0.994565	-0.104122
	1000000010_1111001110	// W0992_2048 = -0.995185	-0.098017
50	1000000010_1111001111	// W0993_2048 = -0.995481	-0.094963
	1000000010_1111010001	// W0994_2048 = -0.995767	-0.091909
	1000000010_1111010100	// W0996_2048 = -0.996313	-0.085797
	1000000010_1111010111	// W0998_2048 = -0.996820	-0.079682
	1000000010_1111011001	// W0999_2048 = -0.997060	-0.076624
55	1000000001_1111011010	// W1000_2048 = -0.997290	-0.073565
	1000000001_1111011101	// W1002_2048 = -0.997723	-0.067444

	1000000001_1111100001	// W1004_2048 = -0.998118	-0.061321
	1000000001_1111100010	// W1005_2048 = -0.998302	-0.058258
	1000000001_1111100100	// W1006_2048 = -0.998476	-0.055195
5	1000000001_1111100111	// W1008_2048 = -0.998795	-0.049068
	1000000000_1111101010	// W1010_2048 = -0.999078	-0.042938
	1000000000_1111101100	// W1011_2048 = -0.999205	-0.039873
	1000000000_1111101101	// W1012_2048 = -0.999322	-0.036807
	1000000000_1111110000	// W1014_2048 = -0.999529	-0.030675
10	1000000000_1111110011	// W1016_2048 = -0.999699	-0.024541
	1000000000_1111110101	// W1017_2048 = -0.999769	-0.021474
	1000000000_1111110111	// W1018_2048 = -0.999831	-0.018407
	1000000000_1111111010	// W1020_2048 = -0.999925	-0.012272
	1000000000_1111111101	// W1022_2048 = -0.999981	-0.006136
15	1000000000_1111111110	// W1023_2048 = -0.999995	-0.003068
	1000000000_0000000011	// W1026_2048 = -0.999981	+0.006136
	1000000000_0000001000	// W1029_2048 = -0.999882	+0.015339
	1000000000_0000001101	// W1032_2048 = -0.999699	+0.024541
	1000000000_0000010001	// W1035_2048 = -0.999431	+0.033741
20	1000000000_0000010110	// W1038_2048 = -0.999078	+0.042938
	1000000001_0000011011	// W1041_2048 = -0.998640	+0.052132
	1000000001_0000011111	// W1044_2048 = -0.998118	+0.061321
	1000000001_0000100100	// W1047_2048 = -0.997511	+0.070505
	1000000010_0000101001	// W1050_2048 = -0.996820	+0.079682
25	1000000010_0000101101	// W1053_2048 = -0.996045	+0.088854
	1000000010_0000110010	// W1056_2048 = -0.995185	+0.098017
	1000000011_0000110111	// W1059_2048 = -0.994240	+0.107172
	1000000011_0000111100	// W1062_2048 = -0.993212	+0.116319
	1000000100_0001000000	// W1065_2048 = -0.992099	+0.125455
30	1000000101_0001000101	// W1068_2048 = -0.990903	+0.134581
	1000000101_0001001010	// W1071_2048 = -0.989622	+0.143695
	1000000110_0001001110	// W1074_2048 = -0.988258	+0.152797
	1000000111_0001010011	// W1077_2048 = -0.986809	+0.161886
	1000001000_0001011000	// W1080_2048 = -0.985278	+0.170962
35	1000001000_0001011100	// W1083_2048 = -0.983662	+0.180023
	1000001001_0001100001	// W1086_2048 = -0.981964	+0.189069
	1000001010_0001100101	// W1089_2048 = -0.980182	+0.198098
	1000001011_0001101010	// W1092_2048 = -0.978317	+0.207111
	1000001100_0001101111	// W1095_2048 = -0.976370	+0.216107
40	1000001101_0001110011	// W1098_2048 = -0.974339	+0.225084
	1000001110_0001111000	// W1101_2048 = -0.972226	+0.234042
	1000001111_0001111100	// W1104_2048 = -0.970031	+0.242980
	1000010001_0010000001	// W1107_2048 = -0.967754	+0.251898
	1000010010_0010000110	// W1110_2048 = -0.965394	+0.260794
45	1000010011_0010001010	// W1113_2048 = -0.962953	+0.269668
	1000010100_0010001111	// W1116_2048 = -0.960431	+0.278520
	1000010110_0010010011	// W1119_2048 = -0.957826	+0.287347
	1000010111_0010011000	// W1122_2048 = -0.955141	+0.296151
	1000011000_0010011100	// W1125_2048 = -0.952375	+0.304929
50	1000011010_0010100001	// W1128_2048 = -0.949528	+0.313682
	1000011011_0010100101	// W1131_2048 = -0.946601	+0.322408
	1000011101_0010101010	// W1134_2048 = -0.943593	+0.331106
	1000011110_0010101110	// W1137_2048 = -0.940506	+0.339777
	1000100000_0010110010	// W1140_2048 = -0.937339	+0.348419
55	1000100010_0010110111	// W1143_2048 = -0.934093	+0.357031
	1000100011_0010111011	// W1146_2048 = -0.930767	+0.365613
	1000100101_0011000000	// W1149_2048 = -0.927363	+0.374164

	1000100111_0011000100	// W1152_2048 = -0.920380	+0.382683
	1000101001_0011001000	// W1155_2048 = -0.920318	+0.391170
	1000101011_0011001101	// W1158_2048 = -0.916679	+0.399624
	1000101101_0011010001	// W1161_2048 = -0.912962	+0.408044
5	1000101111_0011010101	// W1164_2048 = -0.909168	+0.416430
	1000110000_0011011001	// W1167_2048 = -0.905297	+0.424780
	1000110011_0011011110	// W1170_2048 = -0.901349	+0.433094
	1000110101_0011100010	// W1173_2048 = -0.897325	+0.441371
	1000110111_0011100110	// W1176_2048 = -0.893224	+0.449611
10	1000111001_0011101010	// W1179_2048 = -0.889048	+0.457813
	1000111011_0011101111	// W1182_2048 = -0.884797	+0.465976
	1000111101_0011110011	// W1185_2048 = -0.880471	+0.474100
	1000111111_0011110111	// W1188_2048 = -0.876070	+0.482184
	1001000010_0011111011	// W1191_2048 = -0.871595	+0.490226
15	1001000100_0011111111	// W1194_2048 = -0.867046	+0.498228
	1001000110_0100000011	// W1197_2048 = -0.862424	+0.506187
	1001001001_0100000111	// W1200_2048 = -0.857729	+0.514103
	1001001011_0100001011	// W1203_2048 = -0.852961	+0.521975
	1001001110_0100001111	// W1206_2048 = -0.848120	+0.529804
20	1001010000_0100010011	// W1209_2048 = -0.843208	+0.537587
	1001010011_0100010111	// W1212_2048 = -0.838225	+0.545325
	1001010101_0100011011	// W1215_2048 = -0.833170	+0.553017
	1001011000_0100011111	// W1218_2048 = -0.828045	+0.560662
	1001011011_0100100011	// W1221_2048 = -0.822850	+0.568259
25	1001011101_0100100111	// W1224_2048 = -0.817585	+0.575808
	1001100000_0100101011	// W1227_2048 = -0.812251	+0.583309
	1001100011_0100101110	// W1230_2048 = -0.806848	+0.590760
	1001100110_0100110010	// W1233_2048 = -0.801376	+0.598161
	1001101001_0100110110	// W1236_2048 = -0.795837	+0.605511
30	1001101011_0100111010	// W1239_2048 = -0.790230	+0.612810
	1001101110_0100111101	// W1242_2048 = -0.784557	+0.620057
	1001110001_0101000001	// W1245_2048 = -0.778817	+0.627252
	1001110100_0101000101	// W1248_2048 = -0.773010	+0.634393
	1001110111_0101001000	// W1251_2048 = -0.767139	+0.641481
35	1001111010_0101001100	// W1254_2048 = -0.761202	+0.648514
	1001111101_0101010000	// W1257_2048 = -0.755201	+0.655493
	1010000000_0101010011	// W1260_2048 = -0.749136	+0.662416
	1010000100_0101010111	// W1263_2048 = -0.743008	+0.669283
	1010000111_0101011010	// W1266_2048 = -0.736817	+0.676093
40	1010001010_0101011110	// W1269_2048 = -0.730563	+0.682846
	1010001101_0101100001	// W1272_2048 = -0.724247	+0.689541
	1010010000_0101100100	// W1275_2048 = -0.717870	+0.696177
	1010010100_0101101000	// W1278_2048 = -0.711432	+0.702755
	1010010111_0101101011	// W1281_2048 = -0.704934	+0.709273
45	1010011010_0101101110	// W1284_2048 = -0.698376	+0.715731
	1010011110_0101110010	// W1287_2048 = -0.691759	+0.722128
	1010100001_0101110101	// W1290_2048 = -0.685084	+0.728464
	1010100101_0101111000	// W1293_2048 = -0.678350	+0.734739
	1010101000_0101111011	// W1296_2048 = -0.671559	+0.740951
50	1010101100_0101111111	// W1299_2048 = -0.664711	+0.747101
	1010101111_0110000010	// W1302_2048 = -0.657807	+0.753187
	1010110011_0110000101	// W1305_2048 = -0.650847	+0.759209
	1010110110_0110001000	// W1308_2048 = -0.643832	+0.765167
	1010111010_0110001011	// W1311_2048 = -0.636762	+0.771061
55	1010111110_0110001110	// W1314_2048 = -0.629638	+0.776888
	1011000001_0110010001	// W1317_2048 = -0.622461	+0.782651

	1011000101_0110010100	// W1320_2048 = -0.615232	+0.788346
	1011001001_0110010111	// W1323_2048 = -0.607950	+0.793975
	1011001100_0110011001	// W1326_2048 = -0.600616	+0.799537
	1011010000_0110011100	// W1329_2048 = -0.593232	+0.805031
5	1011010100_0110011111	// W1332_2048 = -0.585798	+0.810457
	1011011000_0110100010	// W1335_2048 = -0.578314	+0.815814
	1011011100_0110100100	// W1338_2048 = -0.570781	+0.821103
	1011100000_0110100111	// W1341_2048 = -0.563199	+0.826321
	1011100100_0110101010	// W1344_2048 = -0.555570	+0.831470
10	1011100111_0110101100	// W1347_2048 = -0.547894	+0.836548
	1011101011_0110101111	// W1350_2048 = -0.540171	+0.841555
	1011101111_0110110001	// W1353_2048 = -0.532403	+0.846491
	1011110011_0110110100	// W1356_2048 = -0.524590	+0.851355
	1011110111_0110110110	// W1359_2048 = -0.516732	+0.856147
15	1011111011_0110111001	// W1362_2048 = -0.508830	+0.860867
	1100000000_0110111011	// W1365_2048 = -0.500885	+0.865514
	1100000100_0110111101	// W1368_2048 = -0.492898	+0.870087
	1100001000_0111000000	// W1371_2048 = -0.484869	+0.874587
	1100001100_0111000010	// W1374_2048 = -0.476799	+0.879012
20	1100010000_0111000100	// W1377_2048 = -0.468689	+0.883363
	1100010100_0111000110	// W1380_2048 = -0.460539	+0.887640
	1100011000_0111001001	// W1383_2048 = -0.452350	+0.891841
	1100011101_0111001011	// W1386_2048 = -0.444122	+0.895966
	1100100001_0111001101	// W1389_2048 = -0.435857	+0.900016
25	1100100101_0111001111	// W1392_2048 = -0.427555	+0.903989
	1100101001_0111010001	// W1395_2048 = -0.419217	+0.907886
	1100101110_0111010011	// W1398_2048 = -0.410843	+0.911706
	1100110010_0111010101	// W1401_2048 = -0.402435	+0.915449
	1100110110_0111010111	// W1404_2048 = -0.393992	+0.919114
30	1100111011_0111011000	// W1407_2048 = -0.385516	+0.922701
	1100111111_0111011010	// W1410_2048 = -0.377007	+0.926210
	1101000011_0111011100	// W1413_2048 = -0.368467	+0.929641
	1101001000_0111011110	// W1416_2048 = -0.359895	+0.932993
	1101001100_0111011111	// W1419_2048 = -0.351293	+0.936266
35	1101010001_0111100001	// W1422_2048 = -0.342661	+0.939459
	1101010101_0111100011	// W1425_2048 = -0.334000	+0.942573
	1101011001_0111100100	// W1428_2048 = -0.325310	+0.945607
	1101011110_0111100110	// W1431_2048 = -0.316593	+0.948561
	1101100010_0111100111	// W1434_2048 = -0.307850	+0.951435
40	1101100111_0111101001	// W1437_2048 = -0.299080	+0.954228
	1101101011_0111101010	// W1440_2048 = -0.290285	+0.956940
	1101110000_0111101011	// W1443_2048 = -0.281465	+0.959572
	1101110100_0111101101	// W1446_2048 = -0.272621	+0.962121
	1101111001_0111101110	// W1449_2048 = -0.263755	+0.964590
45	1101111110_0111101111	// W1452_2048 = -0.254866	+0.966976
	1110000010_0111110000	// W1455_2048 = -0.245955	+0.969281
	1110000111_0111110001	// W1458_2048 = -0.237024	+0.971504
	1110001011_0111110011	// W1461_2048 = -0.228072	+0.973644
	1110010000_0111110100	// W1464_2048 = -0.219101	+0.975702
50	1110010100_0111110101	// W1467_2048 = -0.210112	+0.977677
	1110011001_0111110110	// W1470_2048 = -0.201105	+0.979570
	1110011110_0111110110	// W1473_2048 = -0.192080	+0.981379
	1110100010_0111110111	// W1476_2048 = -0.183040	+0.983105
	1110100111_0111111000	// W1479_2048 = -0.173984	+0.984749
55	1110101100_0111111001	// W1482_2048 = -0.164913	+0.986308
	1110110000_0111111010	// W1485_2048 = -0.155828	+0.987784

```

1110110101_0111111010 // W1488_2048 = -0.146730 +0.989177
1110111010_0111111011 // W1491_2048 = -0.137620 +0.990485
1110111110_0111111100 // W1494_2048 = -0.128498 +0.991710
1111000011_0111111100 // W1497_2048 = -0.119365 +0.992850
5 1111001000_0111111101 // W1500_2048 = -0.110222 +0.993907
1111001100_0111111101 // W1503_2048 = -0.101070 +0.994879
1111010001_0111111110 // W1506_2048 = -0.091909 +0.995767
1111010110_0111111110 // W1509_2048 = -0.082740 +0.996571
1111011010_0111111111 // W1512_2048 = -0.073565 +0.997290
10 1111011111_0111111111 // W1515_2048 = -0.064383 +0.997925
1111100100_0111111111 // W1518_2048 = -0.055195 +0.998476
1111101000_0111111111 // W1521_2048 = -0.046003 +0.998941
1111101101_0111111111 // W1524_2048 = -0.036807 +0.999322
1111110010_0111111111 // W1527_2048 = -0.027608 +0.999619
15 1111110111_0111111111 // W1530_2048 = -0.018407 +0.999831
1111111011_0111111111 // W1533_2048 = -0.009204 +0.999958

```

Listing 17

```

20 // 512 point FFT twiddle factor coefficients (Radix 4+2).
// Coefficients stored as non-fractional 10 bit integers (scale 1 ).
// Real Coefficient (cosine value) is coefficient high-byte.
// Imaginary Coefficient (sine value) is coefficient low-byte.

25 0111111111_0000000000 // W0000_0512 = +1.000000 -0.000000
0111111111_1111111010 // W0001_0512 = +0.999925 -0.012272
0111111111_1111110011 // W0002_0512 = +0.999699 -0.024541
0111111111_1111101101 // W0003_0512 = +0.999322 -0.036807
0111111111_1111100111 // W0004_0512 = +0.998795 -0.049068
30 0111111111_1111100001 // W0005_0512 = +0.998118 -0.061321
0111111111_1111011010 // W0006_0512 = +0.997290 -0.073565
0111111110_1111010100 // W0007_0512 = +0.996313 -0.085797
0111111110_1111001110 // W0008_0512 = +0.995185 -0.098017
0111111101_1111001000 // W0009_0512 = +0.993907 -0.110222
35 0111111100_1111000001 // W0010_0512 = +0.992480 -0.122411
0111111011_1110111011 // W0011_0512 = +0.990903 -0.134581
0111111010_1110110101 // W0012_0512 = +0.989177 -0.146730
0111111001_1110101111 // W0013_0512 = +0.987301 -0.158858
0111111000_1110101000 // W0014_0512 = +0.985278 -0.170962
40 0111110111_1110100010 // W0015_0512 = +0.983105 -0.183040
0111110110_1110011100 // W0016_0512 = +0.980785 -0.195090
0111110101_1110010110 // W0017_0512 = +0.978317 -0.207111
0111110100_1110010000 // W0018_0512 = +0.975702 -0.219101
0111110010_1110001010 // W0019_0512 = +0.972940 -0.231058
45 0111110001_1110000100 // W0020_0512 = +0.970031 -0.242980
0111101111_1101111110 // W0021_0512 = +0.966976 -0.254866
0111101101_1101110111 // W0022_0512 = +0.963776 -0.266713
0111101100_1101110001 // W0023_0512 = +0.960431 -0.278520
0111101010_1101101011 // W0024_0512 = +0.956940 -0.290285
50 0111101000_1101100101 // W0025_0512 = +0.953306 -0.302006
0111100110_1101011111 // W0026_0512 = +0.949528 -0.313682
0111100100_1101011001 // W0027_0512 = +0.945607 -0.325310
0111100010_1101010100 // W0028_0512 = +0.941544 -0.336890
0111100000_1101001110 // W0029_0512 = +0.937339 -0.348419
55 0111011110_1101001000 // W0030_0512 = +0.932993 -0.359895
0111011011_1101000010 // W0031_0512 = +0.928506 -0.371317

```


	0111011001_1100111100	// W0032_0512 = +0.923880	-0.382683
	0111010111_1100110110	// W0033_0512 = +0.919114	-0.393992
	0111010100_1100110001	// W0034_0512 = +0.914210	-0.405241
	0111010001_1100101011	// W0035_0512 = +0.909168	-0.416430
5	0111001111_1100100101	// W0036_0512 = +0.903989	-0.427555
	0111001100_1100011111	// W0037_0512 = +0.898674	-0.438616
	0111001001_1100011010	// W0038_0512 = +0.893224	-0.449611
	0111000110_1100010100	// W0039_0512 = +0.887640	-0.460539
	0111000100_1100001111	// W0040_0512 = +0.881921	-0.471397
10	0111000001_1100001001	// W0041_0512 = +0.876070	-0.482184
	0110111101_1100000100	// W0042_0512 = +0.870087	-0.492898
	0110111010_1011111110	// W0043_0512 = +0.863973	-0.503538
	0110110111_1011111001	// W0044_0512 = +0.857729	-0.514103
	0110110100_1011110011	// W0045_0512 = +0.851355	-0.524590
15	0110110001_1011101110	// W0046_0512 = +0.844854	-0.534998
	0110101101_1011101001	// W0047_0512 = +0.838225	-0.545325
	0110101010_1011100100	// W0048_0512 = +0.831470	-0.555570
	0110100110_1011011110	// W0049_0512 = +0.824589	-0.565732
	0110100011_1011011001	// W0050_0512 = +0.817585	-0.575808
20	0110011111_1011010100	// W0051_0512 = +0.810457	-0.585798
	0110011011_1011001111	// W0052_0512 = +0.803208	-0.595699
	0110010111_1011001010	// W0053_0512 = +0.795837	-0.605511
	0110010100_1011000101	// W0054_0512 = +0.788346	-0.615232
	0110010000_1011000000	// W0055_0512 = +0.780737	-0.624859
25	0110001100_1010111011	// W0056_0512 = +0.773010	-0.634393
	0110001000_1010110110	// W0057_0512 = +0.765167	-0.643832
	0110000100_1010110010	// W0058_0512 = +0.757209	-0.653173
	0110000000_1010101101	// W0059_0512 = +0.749136	-0.662416
	0101111011_1010101000	// W0060_0512 = +0.740951	-0.671559
30	0101110111_1010100100	// W0061_0512 = +0.732654	-0.680601
	0101110011_1010011111	// W0062_0512 = +0.724247	-0.689541
	0101101110_1010011010	// W0063_0512 = +0.715731	-0.698376
	0101101010_1010010110	// W0064_0512 = +0.707107	-0.707107
	0101100110_1010010010	// W0065_0512 = +0.698376	-0.715731
35	0101100001_1010001101	// W0066_0512 = +0.689541	-0.724247
	0101011100_1010001001	// W0067_0512 = +0.680601	-0.732654
	0101011000_1010000101	// W0068_0512 = +0.671559	-0.740951
	0101010011_1010000000	// W0069_0512 = +0.662416	-0.749136
	0101001110_1001111100	// W0070_0512 = +0.653173	-0.757209
40	0101001010_1001111000	// W0071_0512 = +0.643832	-0.765167
	0101000101_1001110100	// W0072_0512 = +0.634393	-0.773010
	0101000000_1001110000	// W0073_0512 = +0.624859	-0.780737
	0100111011_1001101100	// W0074_0512 = +0.615232	-0.788346
	0100110110_1001101001	// W0075_0512 = +0.605511	-0.795837
45	0100110001_1001100101	// W0076_0512 = +0.595699	-0.803208
	0100101100_1001100001	// W0077_0512 = +0.585798	-0.810457
	0100100111_1001011101	// W0078_0512 = +0.575808	-0.817585
	0100100010_1001011010	// W0079_0512 = +0.565732	-0.824589
	0100011100_1001010110	// W0080_0512 = +0.555570	-0.831470
50	0100010111_1001010011	// W0081_0512 = +0.545325	-0.838225
	0100010010_1001001111	// W0082_0512 = +0.534998	-0.844854
	0100001101_1001001100	// W0083_0512 = +0.524590	-0.851355
	0100000111_1001001001	// W0084_0512 = +0.514103	-0.857729
	0100000010_1001000110	// W0085_0512 = +0.503538	-0.863973
55	0011111100_1001000011	// W0086_0512 = +0.492898	-0.870087
	0011110111_1000111111	// W0087_0512 = +0.482184	-0.876070

	0011110001_1000111100	// W0088_0512 = -0.471397	-0.881921
	0011101100_1000111010	// W0089_0512 = +0.460539	-0.887640
	0011100110_1000110111	// W0090_0512 = +0.449611	-0.893224
5	0011100001_1000110100	// W0091_0512 = +0.438616	-0.898674
	0011011011_1000110001	// W0092_0512 = +0.427555	-0.903989
	0011010101_1000101111	// W0093_0512 = +0.416430	-0.909168
	0011001111_1000101100	// W0094_0512 = +0.405241	-0.914210
	0011001010_1000101001	// W0095_0512 = +0.393992	-0.919114
10	0011000100_1000100111	// W0096_0512 = +0.382683	-0.923880
	0010111110_1000100101	// W0097_0512 = +0.371317	-0.928506
	0010111000_1000100010	// W0098_0512 = +0.359895	-0.932993
	0010110010_1000100000	// W0099_0512 = +0.348419	-0.937339
	0010101100_1000011110	// W0100_0512 = +0.336890	-0.941544
15	0010100111_1000011100	// W0101_0512 = +0.325310	-0.945607
	0010100001_1000011010	// W0102_0512 = +0.313682	-0.949528
	0010011011_1000011000	// W0103_0512 = +0.302006	-0.953306
	0010010101_1000010110	// W0104_0512 = +0.290285	-0.956940
	0010001111_1000010100	// W0105_0512 = +0.278520	-0.960431
20	0010001001_1000010011	// W0106_0512 = +0.266713	-0.963776
	0010000010_1000010001	// W0107_0512 = +0.254866	-0.966976
	0001111100_1000001111	// W0108_0512 = +0.242980	-0.970031
	0001110110_1000001110	// W0109_0512 = +0.231058	-0.972940
	0001110000_1000001100	// W0110_0512 = +0.219101	-0.975702
25	0001101010_1000001011	// W0111_0512 = +0.207111	-0.978317
	0001100100_1000001010	// W0112_0512 = +0.195090	-0.980785
	0001011110_1000001001	// W0113_0512 = +0.183040	-0.983105
	0001011000_1000001000	// W0114_0512 = +0.170962	-0.985278
	0001010001_1000000111	// W0115_0512 = +0.158858	-0.987301
30	0001001011_1000000110	// W0116_0512 = +0.146730	-0.989177
	0001000101_1000000101	// W0117_0512 = +0.134581	-0.990903
	0000111111_1000000100	// W0118_0512 = +0.122411	-0.992480
	0000111000_1000000011	// W0119_0512 = +0.110222	-0.993907
	0000110010_1000000010	// W0120_0512 = +0.098017	-0.995185
35	0000101100_1000000010	// W0121_0512 = +0.085797	-0.996313
	0000100110_1000000001	// W0122_0512 = +0.073565	-0.997290
	0000011111_1000000001	// W0123_0512 = +0.061321	-0.998118
	0000011001_1000000000	// W0124_0512 = +0.049068	-0.998795
	0000010011_1000000000	// W0125_0512 = +0.036807	-0.999322
40	0000001101_1000000000	// W0126_0512 = +0.024541	-0.999699
	0000000110_1000000000	// W0127_0512 = +0.012272	-0.999925
	0000000000_1000000000	// W0128_0512 = +0.000000	-1.000000
	1111111010_1000000000	// W0129_0512 = -0.012272	-0.999925
	1111110011_1000000000	// W0130_0512 = -0.024541	-0.999699
	1111100111_1000000001	// W0132_0512 = -0.049068	-0.998795
45	1111011010_1000000001	// W0134_0512 = -0.073565	-0.997290
	1111010100_1000000010	// W0135_0512 = -0.085797	-0.996313
	1111001110_1000000010	// W0136_0512 = -0.098017	-0.995185
	1111000001_1000000100	// W0138_0512 = -0.122411	-0.992480
50	1110110101_1000000110	// W0140_0512 = -0.146730	-0.989177
	1110101111_1000000111	// W0141_0512 = -0.158858	-0.987301
	1110101000_1000001000	// W0142_0512 = -0.170962	-0.985278
	1110011100_1000001010	// W0144_0512 = -0.195090	-0.980785
	1110010000_1000001100	// W0146_0512 = -0.219101	-0.975702
55	1110001010_1000001110	// W0147_0512 = -0.231058	-0.972940
	1110000100_1000001111	// W0148_0512 = -0.242980	-0.970031
	1101110111_1000010011	// W0150_0512 = -0.266713	-0.963776

	1101101011_1000010110	// WO152_0512 = -0.290285	-0.956940
	1101100101_1000011000	// WO153_0512 = -0.302006	-0.953306
	1101011111_1000011010	// WO154_0512 = -0.313682	-0.949528
	1101010100_1000011110	// WO156_0512 = -0.336890	-0.941544
5	1101001000_1000100010	// WO158_0512 = -0.359895	-0.932993
	1101000010_1000100101	// WO159_0512 = -0.371317	-0.928506
	1100111100_1000100111	// WO160_0512 = -0.382683	-0.923880
	1100110001_1000101100	// WO162_0512 = -0.405241	-0.914210
	1100100101_1000110001	// WO164_0512 = -0.427555	-0.903989
10	1100011111_1000110100	// WO165_0512 = -0.438616	-0.898674
	1100011010_1000110111	// WO166_0512 = -0.449611	-0.893224
	1100001111_1000111100	// WO168_0512 = -0.471397	-0.881921
	1100000100_1001000011	// WO170_0512 = -0.492898	-0.870087
	1011111110_1001000110	// WO171_0512 = -0.503538	-0.863973
15	1011111001_1001001001	// WO172_0512 = -0.514103	-0.857729
	1011101110_1001001111	// WO174_0512 = -0.534998	-0.844854
	1011100100_1001010110	// WO176_0512 = -0.555570	-0.831470
	1011011110_1001011010	// WO177_0512 = -0.565732	-0.824589
	1011011001_1001011101	// WO178_0512 = -0.575808	-0.817585
20	1011001111_1001100101	// WO180_0512 = -0.595699	-0.803208
	1011000101_1001101100	// WO182_0512 = -0.615232	-0.788346
	1011000000_1001110000	// WO183_0512 = -0.624859	-0.780737
	1010111011_1001110100	// WO184_0512 = -0.634393	-0.773010
	1010110010_1001111100	// WO186_0512 = -0.653173	-0.757209
25	1010101000_1010000101	// WO188_0512 = -0.671559	-0.740951
	1010100100_1010001001	// WO189_0512 = -0.680601	-0.732654
	1010011111_1010001101	// WO190_0512 = -0.689541	-0.724247
	1010010110_1010010110	// WO192_0512 = -0.707107	-0.707107
	1010001101_1010011111	// WO194_0512 = -0.724247	-0.689541
30	1010001001_1010100100	// WO195_0512 = -0.732654	-0.680601
	1010000101_1010101000	// WO196_0512 = -0.740951	-0.671559
	1001111100_1010110010	// WO198_0512 = -0.757209	-0.653173
	1001110100_1010111011	// WO200_0512 = -0.773010	-0.634393
	1001110000_1011000000	// WO201_0512 = -0.780737	-0.624859
35	1001101100_1011000101	// WO202_0512 = -0.788346	-0.615232
	1001100101_1011001111	// WO204_0512 = -0.803208	-0.595699
	1001011101_1011011001	// WO206_0512 = -0.817585	-0.575808
	1001011010_1011011110	// WO207_0512 = -0.824589	-0.565732
	1001010110_1011100100	// WO208_0512 = -0.831470	-0.555570
40	1001001111_1011101110	// WO210_0512 = -0.844854	-0.534998
	1001001001_1011111001	// WO212_0512 = -0.857729	-0.514103
	1001000110_1011111110	// WO213_0512 = -0.863973	-0.503538
	1001000011_1100000100	// WO214_0512 = -0.870087	-0.492898
	1000111100_1100001111	// WO216_0512 = -0.881921	-0.471397
45	1000110111_1100011010	// WO218_0512 = -0.893224	-0.449611
	1000110100_1100011111	// WO219_0512 = -0.898674	-0.438616
	1000110001_1100100101	// WO220_0512 = -0.903989	-0.427555
	1000101100_1100110001	// WO222_0512 = -0.914210	-0.405241
	1000100111_1100111100	// WO224_0512 = -0.923880	-0.382683
50	1000100101_1101000010	// WO225_0512 = -0.928506	-0.371317
	1000100010_1101001000	// WO226_0512 = -0.932993	-0.359895
	1000011110_1101010100	// WO228_0512 = -0.941544	-0.336890
	1000011010_1101011111	// WO230_0512 = -0.949528	-0.313682
	1000011000_1101100101	// WO231_0512 = -0.953306	-0.302006
55	1000010110_1101101011	// WO232_0512 = -0.956940	-0.290285
	1000010011_1101110111	// WO234_0512 = -0.963776	-0.266713

	1000001111_1110000100	// W0236_0512 = -0.970031	-0.242980
	1000001110_1110001010	// W0237_0512 = -0.972940	-0.231058
	1000001100_1110010000	// W0238_0512 = -0.975702	-0.219101
5	1000001010_1110011100	// W0240_0512 = -0.980785	-0.195090
	1000001000_1110101000	// W0242_0512 = -0.985278	-0.170962
	1000000111_1110101111	// W0243_0512 = -0.987301	-0.158858
	1000000110_1110110101	// W0244_0512 = -0.989177	-0.146730
	1000000100_1111000001	// W0246_0512 = -0.992480	-0.122411
	1000000010_1111001110	// W0248_0512 = -0.995185	-0.098017
10	1000000010_1111010100	// W0249_0512 = -0.996313	-0.085797
	1000000001_1111011010	// W0250_0512 = -0.997290	-0.073565
	1000000001_1111100111	// W0252_0512 = -0.998795	-0.049068
	1000000000_1111110011	// W0254_0512 = -0.999699	-0.024541
	1000000000_1111111010	// W0255_0512 = -0.999925	-0.012272
15	1000000000_0000001101	// W0258_0512 = -0.999699	+0.024541
	1000000001_0000011111	// W0261_0512 = -0.998118	+0.061321
	1000000010_0000110010	// W0264_0512 = -0.995185	+0.098017
	1000000101_0001000101	// W0267_0512 = -0.990903	+0.134581
	1000001000_0001011000	// W0270_0512 = -0.985278	+0.170962
20	1000001011_0001101010	// W0273_0512 = -0.978317	+0.207111
	1000001111_0001111100	// W0276_0512 = -0.970031	+0.242980
	1000010100_0010001111	// W0279_0512 = -0.960431	+0.278520
	1000011010_0010100001	// W0282_0512 = -0.949528	+0.313682
	1000100000_0010110010	// W0285_0512 = -0.937339	+0.348419
25	1000100111_0011000100	// W0288_0512 = -0.923880	+0.382683
	1000101111_0011010101	// W0291_0512 = -0.909168	+0.416430
	1000110111_0011100110	// W0294_0512 = -0.893224	+0.449611
	1000111111_0011110111	// W0297_0512 = -0.876070	+0.482184
	1001001001_0100000111	// W0300_0512 = -0.857729	+0.514103
30	1001010011_0100010111	// W0303_0512 = -0.838225	+0.545325
	1001011101_0100100111	// W0306_0512 = -0.817585	+0.575808
	1001101001_0100110110	// W0309_0512 = -0.795837	+0.605511
	1001110100_0101000101	// W0312_0512 = -0.773010	+0.634393
	1010000000_0101010011	// W0315_0512 = -0.749136	+0.662416
35	1010001101_0101100001	// W0318_0512 = -0.724247	+0.689541
	1010011010_0101101110	// W0321_0512 = -0.698376	+0.715731
	1010101000_0101111011	// W0324_0512 = -0.671559	+0.740951
	1010110110_0110001000	// W0327_0512 = -0.643832	+0.765167
	1011000101_0110010100	// W0330_0512 = -0.615232	+0.788346
40	1011010100_0110011111	// W0333_0512 = -0.585798	+0.810457
	1011100100_0110101010	// W0336_0512 = -0.555570	+0.831470
	1011110011_0110110100	// W0339_0512 = -0.524590	+0.851355
	1100000100_0110111101	// W0342_0512 = -0.492898	+0.870087
	1100010100_0111000110	// W0345_0512 = -0.460539	+0.887640
45	1100100101_0111001111	// W0348_0512 = -0.427555	+0.903989
	1100110110_0111010111	// W0351_0512 = -0.393992	+0.919114
	1101001000_0111011110	// W0354_0512 = -0.359895	+0.932993
	1101011001_0111100100	// W0357_0512 = -0.325310	+0.945607
	1101101011_0111101010	// W0360_0512 = -0.290285	+0.956940
50	1101111110_0111101111	// W0363_0512 = -0.254866	+0.966976
	1110010000_0111110100	// W0366_0512 = -0.219101	+0.975702
	1110100010_0111110111	// W0369_0512 = -0.183040	+0.983105
	1110110101_0111111010	// W0372_0512 = -0.146730	+0.989177
	1111001000_0111111101	// W0375_0512 = -0.110222	+0.993907
55	1111011010_0111111111	// W0378_0512 = -0.073565	+0.997290
	1111101101_0111111111	// W0381_0512 = -0.036807	+0.999322

Listing 18

```

5  /*FOLDBEGINS 0 0 "Copyright"*/
   /*****
   Copyright (c) Pioneer Digital Design Centre Limited

10  NAME: pilloc_rtl.v
   PURPOSE: Pilot location

   CREATED:   June 1997  BY: T. Foxcroft

15  MODIFIED:

   USED IN PROJECTS:  cofdm only.

   *****/
20  /*FOLDENDS*/
   /*FOLDBEGINS 0 0 "Defines"*/
   `define FFTSIZE    2048
   `define DATABINS    1705
   `define SCATNUM     45
25  `define SCALEFACTOR64Q 3792  //3x8192/sqrt(42)
   `define SCALEFACTOR16Q 3886  //3x8192/sqrt(10)*2
   `define SCALEFACTORQPS 2172  //3x8192/sqrt(2)*8
   `define AVERAGESF    12'hc49 //0.04x4096x32768/1705 = 3145
   /*FOLDENDS*/
30  module chanest (clk, resync, in_valid, in_data, constellation,
                  u_symbol, us_pilots, uc_pilots, ct_pilots, out_tps, tps_valid,
                  uncorrected_iq,
                  out_valid, outi, outq, c_symbol, incfreq, wrstrb, ramindata,
                  ramoutdata, ramaddr);
35  /*FOLDBEGINS 0 0 "i/o"*/
   input clk, resync, in_valid;
   input [23:0] in_data;
   input [1:0] constellation;
   output u_symbol;
40  output us_pilots, uc_pilots, ct_pilots;
   output out_tps, tps_valid;
   output [23:0] uncorrected_iq;
   output out_valid;
   output [7:0] outi;
45  output [7:0] outq;
   output c_symbol;
   output incfreq;
   output wrstrb;
   output [23:0] ramindata;
50  input [23:0] ramoutdata;
   output [10:0] ramaddr;

   /*FOLDENDS*/
   /*FOLDBEGINS 0 0 "TPS location"*/
55  reg [10:0] tpsloc;
   reg [4:0] tpscount;

```

```

always @(tpscount)
begin
  case(tpscount)
    5    5'b00000: tpsloc = 34;
        5'b00001: tpsloc = 50;
        5'b00010: tpsloc = 209;
        5'b00011: tpsloc = 346;
        5'b00100: tpsloc = 413;
        5'b00101: tpsloc = 569;
    10   5'b00110: tpsloc = 595;
        5'b00111: tpsloc = 688;
        5'b01000: tpsloc = 790;
        5'b01001: tpsloc = 901;
        5'b01010: tpsloc = 1073;
    15   5'b01011: tpsloc = 1219;
        5'b01100: tpsloc = 1262;
        5'b01101: tpsloc = 1286;
        5'b01110: tpsloc = 1469;
        5'b01111: tpsloc = 1594;
    20   default: tpsloc = 1687;
  endcase
end
/*FOLDENDS*/
/*FOLDBEGINS 0 0 "continuous pilot location"*/
25  reg [10:0] contloc;
    reg [5:0] contloccount;
    always @(contloccount)
    begin
      case(contloccount)
    30   6'b000000: contloc = 0;
        6'b000001: contloc = 48;
        6'b000010: contloc = 54;
        6'b000011: contloc = 87;
        6'b000100: contloc = 141;
    35   6'b000101: contloc = 156;
        6'b000110: contloc = 192;
        6'b000111: contloc = 201;
        6'b001000: contloc = 255;
        6'b001001: contloc = 279;
    40   6'b001010: contloc = 282;
        6'b001011: contloc = 333;
        6'b001100: contloc = 432;
        6'b001101: contloc = 450;
        6'b001110: contloc = 483;
    45   6'b001111: contloc = 525;
        6'b010000: contloc = 531;
        6'b010001: contloc = 618;
        6'b010010: contloc = 636;
        6'b010011: contloc = 714;
    50   6'b010100: contloc = 759;
        6'b010101: contloc = 765;
        6'b010110: contloc = 780;
        6'b010111: contloc = 804;
        6'b011000: contloc = 873;
    55   6'b011001: contloc = 888;
        6'b011010: contloc = 918;
      endcase
    end
  end

```

```

        6'b011011: contloc = 939;
        6'b011100: contloc = 942;
        6'b011101: contloc = 969;
        6'b011110: contloc = 984;
5       6'b011111: contloc = 1050;
        6'b100000: contloc = 1101;
        6'b100001: contloc = 1107;
        6'b100010: contloc = 1110;
        6'b100011: contloc = 1137;
10      6'b100100: contloc = 1140;
        6'b100101: contloc = 1146;
        6'b100110: contloc = 1206;
        6'b100111: contloc = 1269;
        6'b101000: contloc = 1323;
15      6'b101001: contloc = 1377;
        6'b101010: contloc = 1491;
        6'b101011: contloc = 1683;
        default: contloc = 1704;
        endcase
20     end
    /*FOLDENDS*/
    /*FOLDBEGINS 0 0 "continuous pilot location"*/
    reg [10:0] contloc [44:0];
    reg [5:0] contloccount;
25     initial
    begin
        contloc[0] = 0; contloc[1] = 48; contloc[2] = 54; contloc[3] = 87; contloc[4] = 141;
        contloc[5] = 156; contloc[6] = 192; contloc[7] = 201; contloc[8] = 255; contloc[9] =
30         279;
        contloc[10] = 282; contloc[11] = 333; contloc[12] = 432; contloc[13] = 450;
        contloc[14] = 483;
        contloc[15] = 525; contloc[16] = 531; contloc[17] = 618; contloc[18] = 636;
        contloc[19] = 714;
        contloc[20] = 759; contloc[21] = 765; contloc[22] = 780; contloc[23] = 804;
35         contloc[24] = 873;
        contloc[25] = 888; contloc[26] = 918; contloc[27] = 939; contloc[28] = 942;
        contloc[29] = 969;
        contloc[30] = 984; contloc[31] = 1050; contloc[32] = 1101; contloc[33] = 1107;
        contloc[34] = 1110;
40         contloc[35] = 1137; contloc[36] = 1140; contloc[37] = 1146; contloc[38] = 1206;
        contloc[39] = 1269;
        contloc[40] = 1323; contloc[41] = 1377; contloc[42] = 1491; contloc[43] = 1683;
        contloc[44] = 1704;
    end */
45     /*FOLDENDS*/
    /*FOLDBEGINS 0 0 "Control vars"*/
    reg [1:0] constell;
    reg resynch;
    reg valid,valid0,valid1,valid2,valid3,valid4,valid5,valid6,valid7,valid8;
50     reg [1:0] whichsymbol;
    reg [1:0] pwhichsymbol;
    reg incwhichsymbol;
    reg [23:0] fftdata;
    reg [10:0] fftcount;
55     reg [10:0] tapcount;
    reg [3:0] count12;

```

```

reg [3:0] dcount12;
reg ramdatavalid;
reg tapinit;
reg tapinit1,tapinit2;
5 reg [7:0] nscat;
reg pilot;
reg tapload; //controls when the taps are loaded
reg tapload2;
reg shiftinnewtap;
10 reg filtgo;
/*FOLDENDS*/
/*FOLDBEGINS 0 0 "Channel Est vars"*/
reg [11:0] tapi [5:0];
reg [11:0] tapq [5:0];
15 reg [27:0] sumi;
reg [27:0] sumq;
reg [11:0] chani;
reg [11:0] chanq;
wire [27:0] chani_;
20 wire [27:0] chanq_;
reg [11:0] idata;
reg [11:0] qdata;
/*FOLDENDS*/
/*FOLDBEGINS 0 0 "RAM vars"*/
25 reg [10:0] ramaddr;
reg [10:0] pilotaddr;
wire [10:0] ramaddr_;
wire [10:0] ramaddrrev_;
reg [23:0] ramindata;
30 wire [23:0] ramoutdata;
reg [23:0] ramout;
reg [23:0] ramot;
reg wrstrb;
reg rwtoggle;
35 reg framedata, framedata0;
reg frav, firstfrav;
reg [23:0] avchannel;
reg [11:0] avchan;
reg avlow;
40 wire [23:0] avchan_;
/*FOLDENDS*/
/*FOLDBEGINS 0 0 "Channel calc vars"*/
reg chan_val;
reg chan_val0,chan_val1,chan_val2,chan_val3,chan_val4,out_valid;
45 reg [23:0] sum;
reg [11:0] sumsq;
reg [11:0] sumsqtemp;
reg [11:0] topreal;
reg [11:0] topimag;
50 reg [7:0] outi;
reg [7:0] outitemp;
reg [5:0] outitem;
reg [7:0] outq;
reg [10:0] prbs;
55 //integer intsumi, intsumq,intsumsq,intouti,intoutq;
/*FOLDENDS*/

```



```

/*FOLDBEGINS 0 0 "uncorrected pilot vars"*/
reg u_symbol;
reg us_pilots;
reg uc_pilots;
5 reg [23:0] uncorrected_iq;
reg [2:0] tps_pilots;
reg [5:0] tpsmajcount;
wire [5:0] tpsmajcount_;
reg ct_pilots;
10 reg out_tps, tps_valid;
reg [1:0] pilotdata;
/*FOLDENDS*/
/*FOLDBEGINS 0 0 "pilot locate vars"*/
wire [1:0] which_symbol;
15 wire [10:0] cpooffset;
wire [10:0] pilotramaddr_;
wire [23:0] pilotramin_;
wire pilotwrstrb_;
wire found_pilots;
20 reg pilotlocated;

/*FOLDENDS*/
/*FOLDBEGINS 0 0 "sync function arrays"*/
reg [11:0] sync0;
25 reg [11:0] sync1;
reg [11:0] sync2;
reg [3:0] syncoffset;
always @(dcount12 or valid1 or valid2)
begin
30   if(valid1 || valid2)
       syncoffset = 4'hc-dcount12;
   else
       syncoffset = dcount12;
/*FOLDBEGINS 0 2 ""*/
35 case(syncoffset)
4'h1:
begin
    sync0 = 4046; sync1 = 272; sync2 = 95;
    end
40 4'h2:
begin
    sync0 = 3899; sync1 = 476; sync2 = 168;
    end
45 4'h3:
begin
    sync0 = 3661; sync1 = 614; sync2 = 217;
    end
4'h4:
begin
50 sync0 = 3344; sync1 = 687; sync2 = 243;
    end
4'h5:
begin
55 sync0 = 2963; sync1 = 701; sync2 = 248;
    end
4'h6:

```

```

begin
sync0 = 2534; sync1 = 665; sync2 = 234;
end
4'h7:
5   begin
sync0 = 2076; sync1 = 590; sync2 = 205;
end
4'h8:
10  begin
sync0 = 1609; sync1 = 486; sync2 = 167;
end
4'h9:
begin
15  sync0 = 1152; sync1 = 364; sync2 = 123;
end
4'ha:
begin
20  sync0 = 722; sync1 = 237; sync2 = 78;
end
default
begin
sync0 = 334; sync1 = 113; sync2 = 36;
end
25  endcase
/*FOLDENDS*/
end
/*FOLDENDS*/
always @(posedge clk)
30  begin
/*FOLDBEGINS 0 2 "Control "*/
constell <= constellation;
resynch <= resynch;
if(resynch)
35  begin
/*FOLDBEGINS 0 2 ""*/
valid    <= 1'b0;
valid0   <= 1'b0;
valid1   <= 1'b0;
40  valid2   <= 1'b0;
valid3   <= 1'b0;
valid4   <= 1'b0;
valid5   <= 1'b0;
valid6   <= 1'b0;
45  valid7   <= 1'b0;
valid8   <= 1'b0;
fftcount <= 11'b0;
ramdatavalid <= 1'b0;
chan_val <= 1'b0;
50  tapinit  <= 1'b0;
tapinit1 <= 1'b0;
tapinit2 <= 1'b0;
rwtoggle <= 1'b0;
/*FOLDENDS*/
55  end
else

```

```

begin
/*FOLDBEGINS 0 2 ""*/
    valid <= in_valid;
    valid0 <= valid&&pilottlocated;
5    valid1 <= valid0;
    valid2 <= valid1;
    valid3 <= valid2;
    valid4 <= valid3;
    valid5 <= valid4;
10    valid6 <= valid5;
    valid7 <= valid6;
    valid8 <= valid7;
    if(valid2)

15        fftcount <= fftcount + 1'b1;
        chan_val <= valid4&&filtgo&&framedata;
        incwhichsymbol <= valid1&&(fftcount == ('FFTSIZE-1));
        if(incwhichsymbol)
            begin
20                rwtoggle <= !rwtoggle;
                tapinit <= 1'b1;
                ramdatavalid <= 1'b1;
            end
        else if(valid6)
25            tapinit <= 1'b0;

        tapinit1 <= tapinit;
        tapinit2 <= tapinit1;
30    /*FOLDENDS*/
    end
    fftdata <= in_data;
    /*FOLDBEGINS 0 0 "frame averager"*/
    if(resynch)
35    begin
        frav <= 1'b0;
        firstfrav <= 1'b0;
    end
    else
40    begin
        if(chan_val&&framedata)
            frav <= 1'b1;
        else if(!framedata&&framedata0)
            frav <= 1'b0;
45    if(chan_val&&framedata&&!frav)
            firstfrav <= 1'b1;
        else if(chan_val)
            firstfrav <= 1'b0;
    /*FOLDBEGINS 0 2 "calculate 0.2 x mean channel amplitude"*/
50    if(chan_val0)
        begin
            if(firstfrav)
                begin
55                avchannel <= avmult(sumsqtemp);
                avchan <= avchan_[11:0];
            end
        end
    end

```

```

        else
            avchannel <= avmult(sumsqtemp) + avchannel;
        end
        /*FOLDENDS*/
5       if(chan_val1)
            avlow <= (sumsqtemp<avchan)? 1:0;

    end
    /*FOLDENDS*/
10    if(resynch)
        begin
            framedata <= 1'b0;

            framedata0 <= 1'b0;
            tapload <= 1'b0;
15        end
        else
            begin
                framedata0 <= framedata;
20                if(incwhichsymbol&&(cpoffset==0))
                    framedata <= 1;
                else if(ramdatavalid&&valid2&&(fftcoun == (cpoffset - 1)))
                    framedata <= 1;
                else if(valid2&&(fftcoun == (cpoffset + `DATABINS)))
25                framedata <= 0;
                    tapload <= framedata;
            end
            filtgo <= ramdatavalid&&( valid2? tapload : filtgo);
            tapload2 <= valid&&tapload&&(count12==11)&&(fftcoun!=0);
30            pilot <= (count12==0);
            dcount12 <= count12;
            shiftinnewtap <= !((nscat == 139)|| (nscat == 140)|| (nscat == 141));

            if(incwhichsymbol)
35            begin
                if(!ramdatavalid)
                    begin
                        whichsymbol <= pwhichsymbol;
                        tapcount <= pwhichsymbol*2'b11 + cpoffset;
40                    end
                else
                    begin
                        whichsymbol <= whichsymbol + 1'b1;
                        tapcount <= {whichsymbol[1]^whichsymbol[0],!whichsymbol[0]}*2'b11 +
45                        cpoffset;
                    end
                end
            else
                if(framedata)
50                begin
                    if(fftcoun==cpoffset)
                        begin
                            /*FOLDBEGINS 0 4 "set up the counters"*/
                            //count12 <= ((4-whichsymbol)&4'b0011)*3;
55                            count12 <= {whichsymbol[1]^whichsymbol[0],whichsymbol[0]}*2'b11;
                            if(valid0)

```

```

        nscat <= 8'b0;
        /*FOLDENDS*/
    end
    else
5      begin
        /*FOLDBEGINS 0 4 ""*/
        if(valid)
        begin
10          count12 <= (count12==11)? 4'b0 : count12 + 1'b1;
          tapcount <= tapcount + 1'b1;
          if(count12==11)

              nscat <= nscat + 1'b1;
              end
15          /*FOLDENDS*/
        end
    end
    else
    begin
20      if(tapinit2&&valid5)
        nscat <= 8'b0;
        if(tapinit)
        begin
25          if(valid3||valid4||valid5&&(whichsymbol==2'b0))
            tapcount <= tapcount + 4'hc;
          else
            if(valid6)
              tapcount <= tapcount +
30          {whichsymbol[1]^whichsymbol[0],whichsymbol[0]}*2'b11 + 1'b1;
          end
        end
        /*FOLDENDS*/
        /*FOLDBEGINS 0 2 "Channel Estimation"*/
        if(tapinit2)
35      begin
        /*FOLDBEGINS 0 4 "Read in first 3 or 4 taps"*/
        if(valid5)
            prbs <= alpha12(alpha(whichsymbol));
        else
40          if(valid6||valid7||(valid8&&(whichsymbol==2'b0)))
            prbs <= alpha12(prbs);
            if(valid5)
            begin
45          tapi[0] <= pseudo(ramout[23:12],1'b1);
          tapi[1] <= pseudo(ramout[23:12],1'b1);
          tapi[2] <= pseudo(ramout[23:12],1'b1);
          tapi[3] <= pseudo(ramout[23:12],1'b1);
          tapq[0] <= pseudo(ramout[11:0], 1'b1);
          tapq[1] <= pseudo(ramout[11:0], 1'b1);
50          tapq[2] <= pseudo(ramout[11:0], 1'b1);
          tapq[3] <= pseudo(ramout[11:0], 1'b1);
        end
        else if( !((whichsymbol!=2'b0)&&valid8))
        begin
55          tapi[5] <= tapi[4];
          tapi[4] <= tapi[3];

```

```

    tapi[3] <= tapi[2];
    tapi[2] <= tapi[1];
    tapi[1] <= tapi[0];
    tapi[0] <= pseudo(ramout[23:12],prbs[0]);
5    tapq[5] <= tapq[4];
    tapq[4] <= tapq[3];
    tapq[3] <= tapq[2];
    tapq[2] <= tapq[1];
    tapq[1] <= tapq[0];
10   tapq[0] <= pseudo(ramout[11:0],prbs[0]);

    end
    /*FOLDENDS*/
end
15   else if(framedata)
    begin
/*FOLDBEGINS 0 4 "update taps in normal op."*/
    if(tapload2)
    begin
20       prbs <= alpha12(prbs);
        tapi[5] <= tapi[4];
        tapi[4] <= tapi[3];
        tapi[3] <= tapi[2];
        tapi[2] <= tapi[1];
25       tapi[1] <= tapi[0];
        if(shiftinnewtap)
            tapi[0] <= pseudo(ramout[23:12],prbs[0]);
            tapq[5] <= tapq[4];
            tapq[4] <= tapq[3];
30         tapq[3] <= tapq[2];
            tapq[2] <= tapq[1];
            tapq[1] <= tapq[0];
            if(shiftinnewtap)
                tapq[0] <= pseudo(ramout[11:0],prbs[0]);
35         end
        /*FOLDENDS*/
/*FOLDBEGINS 0 4 "Channel interpolate"*/
    if(pilot)
    begin
40         if(valid4)
            begin
                chani <= tapi[3];
                chanq <= tapq[3];
            end
45         if(valid3)
            begin
                idata <= ramot[23:12];
                qdata <= ramot[11:0];
            end
50         end
        else
            begin
                if(valid1)
                begin
55         sumi <= mult(tapi[0],sync2) - mult(tapi[1],sync1);
            sumq <= mult(tapq[0],sync2);

```

```

        end
        else if(valid2)
        begin
            sumi <= sumi + mult(tapi[2],sync0);
5           sumq <= sumq + mult(tapq[2],sync0) - mult(tapq[1],sync1);
        end
        else if(valid3)
        begin
10           sumi <= sumi + mult(tapi[3],sync0) - mult(tapi[4],sync1);
            sumq <= sumq + mult(tapq[3],sync0) + 12'h800; //2048 for final round-
            ing
            idata <= ramot[23:12];
            qdata <= ramot[11:0];
15          end
        else if(valid4)
        begin
            chani <= chani_[23:12];
            chanq <= chanq_[23:12];
20          end
        end
        //intsumi = (chani[11])? {20'hffff,chani[11:0]}:chani;
        //intsumq = (chanq[11])? {20'hffff,chanq[11:0]}:chanq;
        //if(chan_val) $display(intsumi*intsumi+intsumq*intsumq);
25        /*FOLDENDS*/
    end
    end
    assign chani_ = sumi + mult(tapi[5],sync2) + 12'h800;
    assign chanq_ = sumq + mult(tapq[5],sync2) - mult(tapq[4],sync1);
30    assign avchan_ = avchannel + 24'h000800;
    /*FOLDENDS*/
/*FOLDBEGINS 0 2 "Calculate channel"*/
always @(posedge clk)
begin
35    if(resynch)
        begin
            chan_val0 <= 1'b0;
            chan_val1 <= 1'b0;
            chan_val2 <= 1'b0;
40            chan_val3 <= 1'b0;
            chan_val4 <= 1'b0;
            out_valid <= 1'b0;
        end
    else
45    begin
        chan_val0 <= chan_val;
        chan_val1 <= chan_val0;
        chan_val2 <= chan_val1;
        chan_val3 <= chan_val2;
50        chan_val4 <= chan_val3;
        //out_valid <= chan_val4;
        out_valid <= chan_val4&&ramdatavalid&&!pilotdata[1];
    end
    if(chan_val)
55    sumsqtemp <= sum[22:11];
    if(chan_val0)

```



```

topreal <= sum[23:12];
if(chan_val1)
topimag <= sum[23:12];
if(chan_val2)
5    sumsq <= sum[23:12];
if(chan_val3)
begin

    outitemp <= divider(topreal,sumsq,(constell==0));
10    outitem <= divplussoft(topreal,sumsq,constell);
end
if(chan_val4)
begin
    outq <= divider(topimag,sumsq,(constell==0));
15    outi <= outitemp;
end
//intouti = (outi[7])? {24'hffff, outi[7:0]}:outi;
//intoutq = (outq[7])? {24'hffff, outq[7:0]}:outq;
//if(chan_val&&ramdatavalid) $display(intsumi);
20 //if(chan_val4&&ramdatavalid) $displayb(outitemp,,outitem);
end
always @(chan_val or chan_val0 or chan_val1 or chani or chanq or constell
        or idata or qdata or sumsqtemp)
begin
25    if(chan_val)
        sum = smult(chani,chani,1) + smult(chanq,chanq,1) + 24'h000400;
    else if(chan_val0)
        sum = smult(idata,chani,1) + smult(qdata,chanq,1) + 24'h000800;
    else if(chan_val1)
30    sum = smult(qdata,chani,1) - smult(idata,chanq,1) + 24'h000800;
    else //chan_val2
        begin
            case(constell)
                2'b00:
35                sum = smult(sumsqtemp,`SCALEFACTORQPS,0) + 24'h000800;
                2'b01:
                    sum = smult(sumsqtemp,`SCALEFACTOR16Q,0) + 24'h000800;
                default:
                    sum = smult(sumsqtemp,`SCALEFACTOR64Q,0) + 24'h000800;
40                endcase
            end
        end
    /*FOLDENDS*/
/*FOLDBEGINS 0 2 "Extract Continual and scattered pilots for Freq + Sampling Error
45 Block"*/
always @(posedge clk)
begin
    if(resynch)
        contloccount <= 6'b0;
50    else
        if(ramdatavalid&&valid2&&(pilotaddr==contloc))
            contloccount <= (contloccount == 44)? 6'b0 : contloccount + 1'b1;
            if(ramdatavalid&&valid2&&((pilotaddr==contloc) || pilot))
                uncorrected_iq <= ramot;
55        uc_pilots <=
            ramdatavalid&&framedata&&(pilotaddr==contloc)&&valid2&&!resynch;

```

```

        us_pilots <= ramdatavalid&&framedata&&pilot&&valid2&&!resynch;
        u_symbol <= !resynch&&ramdatavalid&&(valid2? (pilotaddr==0) : u_symbol);
        // $display(pilotaddr,ramot[23:12],valid2,contloccount,uncorrected_iq[
5      23:12],uncorrected_iq[11:0],uc_pilots,us_pilots);

    end
    /*FOLDENDS*/
    /*FOLDBEGINS 0 2 "Extract TPS pilots "*/
    always @(posedge clk)
10   begin
        if(resynch)
            begin
                tpscount <= 5'b0;
                tps_pilots <= 3'b0;
15      tps_valid <= 1'b0;
                ct_pilots <= 1'b0;
            end
        else
            begin
20      if(ramdatavalid&&valid2&&(pilotaddr==tpsloc))
                tpscount <= (tpscount[4]? 5'b0 : tpscount + 1'b1;
                tps_pilots[0] <= valid2? ramdatavalid&&framedata&&(pilotaddr==tpsloc) :
                    tps_pilots[0];
                tps_pilots[1] <= (chan_val? tps_pilots[0] : tps_pilots[1]);
25      tps_pilots[2] <= tps_pilots[1]&&chan_val3;
                tps_valid <= (tpscount==0)&&tps_pilots[2];
                ct_pilots <= tps_pilots[2];
            end
        if(resynch)
30      tpsmajcount <= 6'b0;
        else
            begin
                if(tps_pilots[2])
                    begin
35      if(tpscount==0)
                        begin
                            tpsmajcount <= 6'b0;
                            out_tps <= tpsmajcount_[5];
                        end
40      else
                            tpsmajcount <= tpsmajcount_;
                        end
            end
        end
        if(resynch)
45      pilotdata <= 2'b0;
        else
            begin
                if(valid2)
50      pilotdata[0] <= ramdatavalid&&framedata&&(
                    (pilotaddr==tpsloc)||
                    (pilotaddr==contloc)||
                    pilot
                );
                pilotdata[1] <= chan_val0? pilotdata[0] : pilotdata[1];
55      end
    end

```

```

        // $display(pilot_addr, ramot[23:12], valid2, contloccount, uncorrected_iq[2
        3:12], uncorrected_iq[11:0], uc_pilots, us_pilots);
        // $display(valid2, pilotdata[0], pilotdata[1], pilotdata[2], ct_pilots, ...,
        out_valid, pilotaddr);
5       end
        assign tpsmajcount_ = tps(topreal[11], tpscount, tpsmajcount);

        /*FOLDENDS*/
        /*FOLDBEGINS 1 2 "pilot locate control"*/
10      always @(posedge clk)
        begin
            if(resynch)
                pilotlocated <= 1'b0;
            else
15          if(found_pilots)
            begin
                pilotlocated <= 1'b1;
                pwhichsymbol <= which_symbol + 2'b10;
            end
20          end
        /*FOLDENDS*/
        /*FOLDBEGINS 0 2 "RAM"*/
        always @(posedge clk)
        begin
25          if(pilotlocated)
            begin
                wrstrb <= !valid0;
                if(valid)
                    ramindata <= fftdata;
30                pilotaddr <= ramaddr_ - cpooffset;
                ramaddr <= rwtoggle? ramaddr_ : ramaddrrev_;
                if(valid5) ramot <= ramout;
            end
            else
35          begin
                /*FOLDBEGINS 0 4 ""*/
                wrstrb <= pilotwrstrb_;
                ramindata <= pilotramin_;
                ramaddr <= pilotramaddr_;
40          /*FOLDENDS*/
            end
            ramout <= ramoutdata;
            end
            assign ramaddr_ = (tapinit || framedata && (valid2 && (count12 == 11))) ? tapcount :
45          fftcount;
            assign ramaddrrev_ =
                {ramaddr_[0], ramaddr_[1], ramaddr_[2], ramaddr_[3], ramaddr_[4], ramaddr_[5],
                ramaddr_[6], ramaddr_[7], ramaddr_[8], ramaddr_[9], ramaddr_[10]};
50          /*FOLDENDS*/
            assign c_symbol = whichsymbol[0];

        /*FOLDBEGINS 0 0 ""*/
        always @(posedge clk)
55      begin

```

```

//$display(chan_val,,framedata,,frav,,firstfrav,,,valid2,,valid4,,out_valid
,,avchannel,,avchan,,sumsqtemp,,,avlow,,chan_val1,,);
//$display(tps_valid,,out_tps,,tpscount,,tps_pilots[2]);
//$display(in_data,,filtgo,,valid4,,tapload,,,nscat,,count12,,fftcount,,incw
5  hichsymbol,,,
  //tapcount,,ramaddr,,wrstrb,,rwtoggle
  //);
  //(resynch,,valid,,fftcount,,ramaddr,,ramindata[23:12],,ramoutdata[23:12],,t
  apinit,,tapinit2,,tapcount,,ramout[23:12],,
10 //tapi[0],,tapi[1],,tapi[2],,tapi[3],,tapi[4],,tapi[5]);
  //$display(tapcount,,tapinit2,,valid4,,valid,,valid2,,wrstrb,,fftcount,,fram
  edata,,count12,,tapi[0],,tapi[1],,tapi[2],,tapi[3],,tapi[4],,tapi[5]);
  //$display(,,,intouti,,intoutq,,out_valid,,,valid4,,valid2,,chan_val,,filt
  go,,framedata,,fftcount,,ramindata[23:12]);
15 //if(whichsymbol==1)
  $display(tapinit,,tapcount,,fftcount,,ramindata[23:12],,,,tapcount,,tapi[0]
  ,,tapi[1],,tapi[2],,tapi[3],,tapi[4],,tapi[5],,intsumi,,intsumq,,idata,,qdata);
  //$display(framedata,,pilotaddr,,fftcount,,tapcount,,ramaddr,,ramout[23:12],
  ,ramindata[23:12],,prbs,,us_pilots,,uc_pilots,,ct_pilots,,out_valid,,,contl occount,,
20 //tps_pilots[0],,tps_pilots[1],,tps_pilots[2]);
end
/*FOLDENDS*/
piloc piloc (.clk(clk), .resync(resync), .in_valid(in_valid), .in_data(in_data),
.found_pilots(found_pilots), .which_symbol(which_symbol),
25 .cpoffset(cpooffset), .incfreq(incfreq),
  .ramaddr(pilotramaddr_) , .ramin(pilotramin_) , .ramout(ramout),
  .wrstrb(pilotwrstrb_));
/*FOLDBEGINS 0 2 "functions"*/
/*FOLDBEGINS 0 0 "tps demod"*/
30 function [5:0] tps;
  input tpssign;
  input [4:0] tpscount;
  input [5:0] tpsmajcount;
  reg tpsflip;
35 begin
  case(tpscount)
    5'b00001,5'b00011,5'b00100,5'b00110,5'b01011,5'b01110:
      tpsflip = 0; //added1 since tpscount already incremented
      default:
40      tpsflip = 1;
      endcase
      tps = (tpsflip^tpssign)? tpsmajcount - 1'b1 : tpsmajcount + 1'b1;
  end
endfunction
45 /*FOLDENDS*/
/*FOLDBEGINS 0 0 "pseudo function"*/

function [11:0] pseudo;
input [11:0] data;
50 input flip;
begin
  pseudo = flip? ~data + 1'b1 : data;
end
endfunction
55 /*FOLDENDS*/
/*FOLDBEGINS 0 0 "averager multiplier"*/

```

```

        function [11:0] avmult;
        input [11:0] i;
        reg [23:0] res;
        begin
5         res = (i*AVERAGESF) + 23'h000800; //multiply and round
          avmult = res[23:12];
        end
    endfunction
    /*FOLDENDS*/
10    /*FOLDBEGINS 0 0 "filter tap multiplier"*/
        function [27:0] mult;
        input [11:0] i;
        input [11:0] j;
        reg [23:0] res;
15        reg [11:0] modi;
        reg [11:0] invi;
        begin
            invi = ~i + 1'b1;
            modi = i[11]? invi : i;
20            res = (modi*j); //multiply and round
            mult = i[11]? {4'hf,~res} + 1'b1 : res;
        end
    endfunction
    /*FOLDENDS*/
25    /*FOLDBEGINS 0 0 "signed multiplier"*/
        function [23:0] smult;
        input [11:0] i;
        input [11:0] j;
        input signedj;
30        reg [23:0] res;
        reg [11:0] modi;
        reg [11:0] modj;
        begin
            modi = i[11]? ~i + 1'b1 : i;
35            modj = (j[11]&&signedj)? ~j + 1'b1 : j;
            res = (modi*modj);
            smult = (i[11]^(j[11]&&signedj))? ~res + 1'b1 : res;
        end
    endfunction
40    /*FOLDENDS*/
    /*FOLDBEGINS 0 0 "divider function"*/
        function [7:0] divider;
        input [11:0] dividend;
        input [11:0] divisor;
45        input qpsk;

        reg [11:0] moddividend;
        reg signresult;
        reg [12:0] intval;
50        reg [12:0] carry;
        reg [7:0] divide;
        reg [8:0] signeddivide;
        integer i;
        begin
55            signresult = dividend[11];
            moddividend = dividend[11]? ~dividend + 1'b1 : dividend;

```

```

        divide = 0;
        carry = qpsk? {1'b0,moddividend}:{moddividend,1'b0};
/*FOLDBEGINS 0 2 ""*/
        for(i=0;i<8;i=i+1)
5         begin
            intval = carry - divisor;
            divide[7-i] = !intval[12];
            carry = (intval[12])? {carry[11:0],1'b0} : {intval[11:0],1'b0};
        end
10        /*FOLDENDS*/
        //signeddivide = signresult? ~divide + 2'b10 : divide + 1'b1;
        signeddivide = signresult? {1'b1,~divide} + 2'b10 : {1'b0,divide} + 1'b1;
        //$displayb(signeddivide,,divide,,signresult,,constellation,,);
        divider = signeddivide[8:1];
15    end
    endfunction
    /*FOLDENDS*/
    /*FOLDBEGINS 0 0 "divider function with soft decisions added"*/
    function [5:0] divplussoft;
20    input [11:0] dividend;
    input [11:0] divisor;
    input [1:0] constellation;
    reg [11:0] moddividend;
    reg signresult;
25    reg [12:0] intval;
    reg [12:0] carry;
    reg [8:0] divide;
    reg [10:0] signeddivide;
    reg [11:0] fracdivide;
30    integer i;
    begin
        signresult = dividend[11];
        moddividend = dividend[11]? ~dividend + 1'b1 : dividend;
        divide = 0;
35        carry = (constellation==0)? {1'b0,moddividend}:{moddividend,1'b0};
        /*FOLDBEGINS 0 2 ""*/
        for(i=0;i<9;i=i+1)
            begin
                intval = carry - divisor;
40                divide[8-i] = !intval[12];
                carry = (intval[12])? {carry[11:0],1'b0} : {intval[11:0],1'b0};
            end
            /*FOLDENDS*/
            signeddivide = signresult? {2'b11,~divide} + 1'b1 : {2'b0,divide};
45        //$displayb(signeddivide,,divide,,signresult,,constellation,,);
        /*FOLDBEGINS 0 2 "qpsk"*/
        if(constellation==2'b0)
            begin
50                //$writeh(,,signeddivide,,);
                signeddivide = signeddivide + 8'h80;
                //$writeh(signeddivide,,);
                if(signeddivide[10])
                    fracdivide = 9'h0;
55                else
                    if(signeddivide[9]||signeddivide[8])

```

```

        fracdivide = 12'h700;
        else
        begin
5          fracdivide = signeddivide[7:0] + {signeddivide[7:0],1'b0} +
            {signeddivide[7:0],2'b0}; /*7
          fracdivide = fracdivide + 8'h80;
        end
        divplussoft = {3'b0,fracdivide[10:8]};
10      end
      else
        /*FOLDENDS*/
        /*FOLDBEGINS 0 2 "16qam"*/
        if(constellation==2'b01)
        begin
15          $writeh(.,signeddivide,...);
          signeddivide = signeddivide + 8'hc0;
          $writeh(.,signeddivide,...);
          if(signeddivide[10])
          begin
20            signeddivide = 10'b0;
            fracdivide = 9'h0;
          end
          else
          if(signeddivide[9] || (signeddivide[8:7]==2'b11))
25          begin
            fracdivide = 12'h380;
            signeddivide = 10'h100;
          end
          else
          begin
30            fracdivide = signeddivide[6:0] + {signeddivide[6:0],1'b0} +
              {signeddivide[6:0],2'b0}; /*7
            fracdivide = fracdivide + 8'h40;
          end
          divplussoft = {1'b0,signeddivide[8:7],fracdivide[9:7]};
35        end
      /*FOLDENDS*/
      /*FOLDBEGINS 0 2 "32qam"*/
      else
40      begin
        signeddivide = signeddivide + 8'he0;
        if(signeddivide[10])
        begin
          signeddivide = 10'b0;
45          fracdivide = 9'h0;
        end
        else
        if(signeddivide[9] || (signeddivide[8:6]==3'b111))
50        begin
          signeddivide = 10'h180;
          fracdivide = 9'h1c0;
        end
        else
65      begin

```



```

        fracdivide = signeddivide[5:0] + {signeddivide[5:0],1'b0} +
        {signeddivide[5:0],2'b0}; /**7
        fracdivide = fracdivide + 8'h20;
    end
5      divplussoft = {signeddivide[8:6],fracdivide[8:6]};
    end
    /*FOLDENDS*/
end
endfunction
10 /*FOLDENDS*/
/*FOLDBEGINS 0 0 "PRBS alpha3/6/9/12 multiplier"*/
function [10:0] alpha;
input [1:0] which_symbol;
begin
15     case(which_symbol)
        2'b0:
            alpha = 11'b111111111111;
        2'b01:
            alpha = 11'b000111111111;
20     2'b10:
            alpha = 11'b000000111111;
        2'b11:
            alpha = 11'b000000000111;
        endcase
25     end
endfunction
/*FOLDENDS*/
/*FOLDBEGINS 0 0 "PRBS alpha12 multiplier"*/
function [10:0] alpha12;
30 input [10:0] prbsin;
    reg [10:0] prbs0;
    reg [10:0] prbs1;
    reg [10:0] prbs2;
    reg [10:0] prbs3;
35    reg [10:0] prbs4;
    reg [10:0] prbs5;
    reg [10:0] prbs6;
    reg [10:0] prbs7;
    reg [10:0] prbs8;
40    reg [10:0] prbs9;
    reg [10:0] prbs10;
    begin
        prbs0 = {prbsin[0] ^ prbsin[2],prbsin[10:1]};
45        prbs1 = {prbs0[0] ^ prbs0[2],prbs0[10:1]};
        prbs2 = {prbs1[0] ^ prbs1[2],prbs1[10:1]};
        prbs3 = {prbs2[0] ^ prbs2[2],prbs2[10:1]};
        prbs4 = {prbs3[0] ^ prbs3[2],prbs3[10:1]};
        prbs5 = {prbs4[0] ^ prbs4[2],prbs4[10:1]};
50        prbs6 = {prbs5[0] ^ prbs5[2],prbs5[10:1]};
        prbs7 = {prbs6[0] ^ prbs6[2],prbs6[10:1]};
        prbs8 = {prbs7[0] ^ prbs7[2],prbs7[10:1]};
        prbs9 = {prbs8[0] ^ prbs8[2],prbs8[10:1]};
        prbs10 = {prbs9[0] ^ prbs9[2],prbs9[10:1]};
55        alpha12 = {prbs10[0] ^ prbs10[2],prbs10[10:1]};
    end
end

```

```

    endfunction
    /*FOLDENDS*/
    /*FOLDENDS*/
endmodule

```

5

Listing 19

```

10  /*FOLDBEGINS 0 0 "Copyright"*/
    /******
    Copyright (c) Pioneer Digital Design Centre Limited

    NAME: pilloc_rtl.v
15  PURPOSE: Pilot location

    CREATED:   June 1997 BY: J. Parker (C code)

20  MODIFIED:   BY: T. Foxcroft

    USED IN PROJECTS: cofdm only.

    /******/
25  /*FOLDENDS*/
    `define FFTSIZE 2048
    `define SCATNUM 45
    module pilloc (clk, resync, in_valid, in_data, found_pilots, which_symbol, cpoffset,
    incfreq,
30          ramaddr, rammin, ramout, wrstrb);
        /*FOLDBEGINS 0 0 "i/o"*/
        input clk, resync, in_valid;
        input [23:0] in_data;
        output found_pilots;
35        output [1:0] which_symbol;
        output [10:0] cpoffset;
        output incfreq;
        /*FOLDENDS*/

40        /*FOLDBEGINS 0 0 "ram i/o"*/
        output [10:0] ramaddr;
        reg [10:0] ramaddr_;
        output [23:0] rammin;
        input [23:0] ramout;
45        output wrstrb;
        reg [10:0] ramaddr;
        reg [23:0] rammin;
        reg wrstrb;
        /*FOLDENDS*/
50        /*FOLDBEGINS 0 0 "vars"*/
        reg found_pilots;
        reg [1:0] which_symbol;
        reg [1:0] which_symbolcount;
        reg [1:0] which_symbol_;
55        reg [10:0] cpoffset;
        reg incfreq;

```

```

reg found_pilot;
reg [19:0] v;
reg [19:0] sum;
reg [3:0] splocoffset;
5 wire [10:0] carrier_number;
reg [10:0] continual_pilot_offset;

reg resynch;
reg [3:0] valid;
10 reg [23:0] fftdata;
reg [10:0] fftcount;
reg contcomplete;
reg firstcontsearch;
reg finishedsearch;
15 reg [4:0] firstscatcomplete;
reg [4:0] failedtolock;
reg [2:0] spmax;
reg [2:0] spmaxfirst;
reg [10:0] pilot_offset;
20 reg [1:0] sploc1zero;
reg [10:0] sploc0;
reg [5:0] sploc1;
reg [10:0] splocmaxcount;

25 reg [3:0] spoffset;
reg [19:0] sumscat [11:0];
reg [19:0] sumscatmax;
reg [3:0] sumscatmaxno0;
reg [3:0] sumscatmaxno1;
30 wire [19:0] sumscat1;
wire [19:0] sumscat3;
wire [19:0] sumscat5;
reg [11:0] sumscatfirst;
reg [4:0] fftfinished;
35 reg ramwritestop; //botch for development purposes
wire [3:0] mod12fftcount;
/*FOLDENDS*/
/*FOLDBEGINS 0 0 "continuous pilot location"*/
reg [10:0] contloc;
40 always @(sploc1)
begin
    case(sploc1)
        6'b000000: contloc = 0;
        6'b000001: contloc = 48;
45 6'b000010: contloc = 54;
        6'b000011: contloc = 87;
        6'b000100: contloc = 141;
        6'b000101: contloc = 156;
        6'b000110: contloc = 192;
50 6'b000111: contloc = 201;
        6'b001000: contloc = 255;
        6'b001001: contloc = 279;
        6'b001010: contloc = 282;
        6'b001011: contloc = 333;
55 6'b001100: contloc = 432;
        6'b001101: contloc = 450;
    endcase
end

```

```

        6'b001110: contloc = 483;
        6'b001111: contloc = 525;
        6'b010000: contloc = 531;
        6'b010001: contloc = 618;
5       6'b010010: contloc = 636;
        6'b010011: contloc = 714;
        6'b010100: contloc = 759;
        6'b010101: contloc = 765;
        6'b010110: contloc = 780;
10      6'b010111: contloc = 804;
        6'b011000: contloc = 873;
        6'b011001: contloc = 888;
        6'b011010: contloc = 918;
        6'b011011: contloc = 939;
15      6'b011100: contloc = 942;
        6'b011101: contloc = 969;
        6'b011110: contloc = 984;
        6'b011111: contloc = 1050;
        6'b100000: contloc = 1101;
20      6'b100001: contloc = 1107;
        6'b100010: contloc = 1110;
        6'b100011: contloc = 1137;
        6'b100100: contloc = 1140;
        6'b100101: contloc = 1146;
25      6'b100110: contloc = 1206;
        6'b100111: contloc = 1269;
        6'b101000: contloc = 1323;
        6'b101001: contloc = 1377;
        6'b101010: contloc = 1491;
30      6'b101011: contloc = 1683;
        default: contloc = 1704;
        endcase
    end
/*FOLDENDS*/
35
    always @(posedge clk)
    begin
        resynch <= resync;
        if(resynch)
40      begin
            valid      <= 4'b0;
            fftcount   <= 11'b0;
            firstscatcomplete <= 5'b0;
            sum        <= 20'b0;
45      sploc0      <= 11'b0;
            sploc1     <= 6'b0;
            contcomplete <= 1'b0;
            failedtolock <= 5'b0;
            spmax      <= 1'b0;
50      spmaxfirst  <= 1'b0;
            ramwritestop <= 1'b0;
            found_pilots <= 1'b0;
            found_pilot <= 1'b0;
            firstcontsearch <= 1'b0;
55      finishedsearch <= 1'b0;
            which_symbolcount <= 2'b0;

```

```

        incfreq    <= 1'b0;
    end
    else
    begin
5      incfreq <= !failedtolock[1]&&failedtolock[0]&&fftfinished[4];
        found_pilots <= !found_pilot&&finishedsearch;
        found_pilot <= finishedsearch;
        valid[0] <= in_valid;
        valid[1] <= valid[0];
10     valid[2] <= valid[1];
        valid[3] <= valid[2];
        fftdata <= in_data;
        if(valid[0]&&!finishedsearch)
15         fftcount <= fftcount + 1'b1;
        //if(fftfinished[0])
        // $display("frame",,fftcount);
        //if(incfreq)
        // $display("tweek");

20    /*FOLDBEGINS 0 4 "locate continual pilots"*/
        spmax[1] <= spmax[0];
        spmax[2] <= spmax[1];
        spmaxfirst[1] <= spmaxfirst[0];
        spmaxfirst[2] <= spmaxfirst[1];
25    //if(fftfinished[3])
        // $display(spoffset,,which_symbol);

        if(fftfinished[3])
        begin
30         failedtolock[1] <= failedtolock[0];
            failedtolock[2] <= failedtolock[1];
            failedtolock[3] <= failedtolock[2];
            failedtolock[4] <= failedtolock[3];

35         if(failedtolock[0])
            begin
            /*FOLDBEGINS 0 2 ""*/
            if(failedtolock[4])
40                 failedtolock[0] <= 1'b0;
                    firstscatcomplete <= 5'b0;
                    ramwritestop <= 1'b0;
                    firstcontsearch <= 1'b0;
                    /*FOLDENDS*/
            end
45         else
            begin
            /*FOLDBEGINS 0 4 ""*/
            firstscatcomplete[0] <= 1'b1;
            firstcontsearch <= !firstscatcomplete[0];
50         ramwritestop <= !ramwritestop||finishedsearch;
            contcomplete <= ramwritestop;
            if(!finishedsearch&&firstscatcomplete[0]&&ramwritestop)
            begin
55                 finishedsearch <= firstcontsearch? 1'b0 :
                    (cpoffset==continual_pilot_offset);
                    cpoffset <= continual_pilot_offset;

```

```

        failedtolock[0] <= !firstcontsearch&&(cpoffset!=continual_pilot_offset);
    end
    /*FOLDENDS*/
5    end
    end
    else
    begin
        firstscatcomplete[1] <= firstscatcomplete[0]&&!contcomplete;
        firstscatcomplete[2] <= firstscatcomplete[1];
10    if(firstscatcomplete[0]&&!finishedsearch&&!contcomplete&&!finishedsearch
        &&(sploc1==44)&&(sploc0==splocmaxcount))
        contcomplete <= 1'b1;
    end
    if(found_pilots)
15    $display(which_symbol,,cpoffset,,spoffset);
        //$display(sum,,contcomplete,,ramwritestop,,which_symbol,,spoffset,,sploc0,,splocmaxcount,,v,,,,,fftfinished[3],,finishedsearch);
        //$display(fftcount,,firstscatcomplete[0],,ramwritestop,,spoffset,,sumsca
        tmaxno1,,,finishedsearch,,found_pilots,,
20    //,,,,,
        //pilot_offset,,which_symbol,,,cpoffset,,failedtolock );
        sploc1zero[0] <= (sploc1 == 0);
        sploc1zero[1] <= sploc1zero[0];

25    if(firstscatcomplete[0]&&!finishedsearch&&!contcomplete&&!finishedsearch)
        begin
            if(sploc1==44)

            begin
30    /*FOLDBEGINS 0 4 ""*/
            //$display(sploc0,,splocmaxcount);
            pilot_offset <= sploc0 + splocoffset;
            which_symbol <= which_symbol_ - which_symbolcount;
            if(sploc0==splocmaxcount)
35    begin
                sploc0 <= 11'b0;
                //contcomplete <= 1'b1;
                which_symbolcount <= 2'b0;
            end
            else
40    begin
                sploc0 <= sploc0 + 2'b11;
                which_symbolcount <= which_symbolcount + 1'b1;
            end
            if(sploc0==0)
45    if(sploc0==0)
                spmaxfirst[0] <= 1'b1;
                sploc1 <= 6'b0;
                spmax[0] <= 1'b1;
                /*FOLDENDS*/
50    end
            else
            begin
                /*FOLDBEGINS 0 4 ""*/
                sploc1 <= sploc1 + 1'b1;
55    spmax[0] <= 1'b0;
                spmaxfirst[0] <= 1'b0;

```

```

/*FOLDENDS*/
    end
    end
    if(firstscatcomplete[2])
5      begin
        if(sploc1zero[1])
            sum <= modulus(ramout[23:12],ramout[11:0]);
        else
10       sum <= modulus(ramout[23:12],ramout[11:0]) + sum;
        end
    /*FOLDENDS*/
    end
/*FOLDBEGINS 0 2 "search for largest continous pilot correlation"*/
    if(spmax[2])
15     begin
        if(spmaxfirst[2])
            begin
                v <= sum;
                continual_pilot_offset <= pilot_offset;
20            end
        else
            begin
                if(sum>v)
                    begin
25                     v <= sum;
                     continual_pilot_offset <= pilot_offset;

                    end
                end
30         //$display(sum,,continual_pilot_offset,,contcomplete,,ramwritestop,,which
            _symbol,,spoffset,,,sploc0,,splocmaxcount,,v);
            //$display(sum);
        end
    /*FOLDENDS*/
35     end
    assign carrier_number = contloc + sploc0 + splocoffset;
/*FOLDBEGINS 0 0 "scattered pilot offset mod 3"*/
    always @(spoffset)
        begin
40         splocoffset = 2'b0;
            splocmaxcount = 342;
            which_symbol_ = 2'b0;
            case(spoffset)
                4'b0000,4'b0011,4'b0110,4'b1001:
45                 begin
                    splocoffset = 2'b0;
                    splocmaxcount = 342;
                    end
                4'b0001,4'b0100,4'b0111,4'b1010:
50                 begin
                    splocoffset = 2'b01;
                    splocmaxcount = 339;
                    end
                4'b0010,4'b0101,4'b1000,4'b1011:
55                 default:
                    begin

```


2

```

        splocoffset = 2'b10;
        splocmaxcount = 339;
    end
endcase
5   case(spoffset)
    4'b0000,4'b0001,4'b0010:
        which_symbol_ = 2'b0;
    4'b0011,4'b0100,4'b0101:
        which_symbol_ = 2'b01;
10   4'b0110,4'b0111,4'b1000:
        which_symbol_ = 2'b10;
        //4'b1001,4'b1010,4'b1011:
        default:
            which_symbol_ = 2'b11;
15   endcase
    end
/*FOLDENDS*/
/*FOLDBEGINS 1 0 "Search for scattered pilots"*/
20   always @(posedge clk)
    begin

        if(resynch)
            sumscatfirst <= 12'hfff;
        else
25         begin
            if(valid[0]&&!finishedsearch)
/*FOLDBEGINS 1 2 "do the accumulations"*/
            case(mod12fftcnt)
30         4'h0:
            begin
                sumscat[0] <= (sumscatfirst[0])? modulus(fftdata[23:12],fftdata[11:0]) :
                sumscat[0] + modulus(fftdata[23:12],fftdata[11:0]);
                sumscatfirst[0] <= 1'b0;
35         end
            4'h1:
            begin
                sumscat[1] <= (sumscatfirst[1])? modulus(fftdata[23:12],fftdata[11:0]) :
                sumscat[1] + modulus(fftdata[23:12],fftdata[11:0]);
40         sumscatfirst[1] <= 1'b0;
            end
            4'h2:
            begin
                sumscat[2] <= (sumscatfirst[2])? modulus(fftdata[23:12],fftdata[11:0]) :
45         sumscat[2] + modulus(fftdata[23:12],fftdata[11:0]);
                sumscatfirst[2] <= 1'b0;
            end
            4'h3:
            begin
50         sumscat[3] <= (sumscatfirst[3])? modulus(fftdata[23:12],fftdata[11:0]) :
                sumscat[3] + modulus(fftdata[23:12],fftdata[11:0]);
                sumscatfirst[3] <= 1'b0;
            end
            4'h4:
55         begin

```

```

        sumscat[4] <= (sumscatfirst[4])? modulus(fftdata[23:12],fftdata[11:0]) :
        sumscat[4] + modulus(fftdata[23:12],fftdata[11:0]);
        sumscatfirst[4] <= 1'b0;
    end
5   4'h5:
    begin
        sumscat[5] <= (sumscatfirst[5])? modulus(fftdata[23:12],fftdata[11:0]) :
        sumscat[5] + modulus(fftdata[23:12],fftdata[11:0]);
        sumscatfirst[5] <= 1'b0;
10  end
    4'h6:
    begin
        sumscat[6] <= (sumscatfirst[6])? modulus(fftdata[23:12],fftdata[11:0]) :
        sumscat[6] + modulus(fftdata[23:12],fftdata[11:0]);
15  sumscatfirst[6] <= 1'b0;
    end
    4'h7:
    begin
        sumscat[7] <= (sumscatfirst[7])? modulus(fftdata[23:12],fftdata[11:0]) :
20  sumscat[7] + modulus(fftdata[23:12],fftdata[11:0]);
        sumscatfirst[7] <= 1'b0;
    end
    4'h8:
    begin
25
        sumscat[8] <= (sumscatfirst[8])? modulus(fftdata[23:12],fftdata[11:0]) :
        sumscat[8] + modulus(fftdata[23:12],fftdata[11:0]);
        sumscatfirst[8] <= 1'b0;
    end
30  4'h9:
    begin
        sumscat[9] <= (sumscatfirst[9])? modulus(fftdata[23:12],fftdata[11:0]) :
        sumscat[9] + modulus(fftdata[23:12],fftdata[11:0]);
        sumscatfirst[9] <= 1'b0;
35  end
    4'ha:
    begin
        sumscat[10] <= (sumscatfirst[10])? modulus(fftdata[23:12],fftdata[11:0]) :
        sumscat[10] + modulus(fftdata[23:12],fftdata[11:0]);
40  sumscatfirst[10] <= 1'b0;
    end
    default:
    begin
        sumscat[11] <= (sumscatfirst[11])? modulus(fftdata[23:12],fftdata[11:0]) :
45  sumscat[11] + modulus(fftdata[23:12],fftdata[11:0]);
        sumscatfirst[11] <= 1'b0;
    end
    endcase
    /*FOLDENDS*/
50  else if(fftfinished[0])
        sumscatfirst <= 12'hfff;
    end
    /*FOLDBEGINS 1 0 "Find offset"*/
    if(resynch)
55  fftfinished <= 5'b0;
    else

```

```

begin
  fftfinished[0] <= valid[0]&&!finishedsearch&&(fftcount==2047);
  fftfinished[1] <= fftfinished[0];
  fftfinished[2] <= fftfinished[1];
5   fftfinished[3] <= fftfinished[2];
   fftfinished[4] <= fftfinished[3];
end
if(!ramwritestop)
begin
10   if(fftfinished[0])
      begin
        sumscat[0] <= (sumscat[0] > sumscat[1])? sumscat[0] : sumscat[1];
        sumscat[1] <= (sumscat[0] > sumscat[1])? 0 : 1;
        sumscat[2] <= (sumscat[2] > sumscat[3])? sumscat[2] : sumscat[3];
15   sumscat[3] <= (sumscat[2] > sumscat[3])? 2 : 3;
        sumscat[4] <= (sumscat[4] > sumscat[5])? sumscat[4] : sumscat[5];
        sumscat[5] <= (sumscat[4] > sumscat[5])? 4 : 5;
        sumscat[6] <= (sumscat[6] > sumscat[7])? sumscat[6] : sumscat[7];
        sumscat[7] <= (sumscat[6] > sumscat[7])? 6 : 7;
20   sumscat[8] <= (sumscat[8] > sumscat[9])? sumscat[8] : sumscat[9];
        sumscat[9] <= (sumscat[8] > sumscat[9])? 8 : 9;
        sumscat[10] <= (sumscat[10] > sumscat[11])? sumscat[10] : sumscat[11];
        sumscat[11] <= (sumscat[10] > sumscat[11])? 10 : 11;

25   end
      if(fftfinished[1])
        begin
          sumscat[0] <= (sumscat[0] > sumscat[2])? sumscat[0] : sumscat[2];
          sumscat[1] <= (sumscat[0] > sumscat[2])? sumscat[1] : sumscat[3];
30   sumscat[2] <= (sumscat[4] > sumscat[6])? sumscat[4] : sumscat[6];
          sumscat[3] <= (sumscat[4] > sumscat[6])? sumscat[5] : sumscat[7];
          sumscat[4] <= (sumscat[8] > sumscat[10])? sumscat[8] : sumscat[10];
          sumscat[5] <= (sumscat[8] > sumscat[10])? sumscat[9] : sumscat[11];

          end
        if(fftfinished[2]&&!ramwritestop)
          spoffset <= sumscatmaxno1;
          end
          if(fftfinished[0])
            begin
40   $display(sumscat[0]);
          $display(sumscat[1]);
          $display(sumscat[2]);
          $display(sumscat[3]);
          $display(sumscat[4]);
          $display(sumscat[5]);
45   $display(sumscat[6]);
          $display(sumscat[7]);
          $display(sumscat[8]);
          $display(sumscat[9]);
          $display(sumscat[10]);
50   $display(sumscat[11]);
          $display();
            end
          end
55   end

```

```

always @(sumscat[0] or sumscat[1] or sumscat[2] or sumscat[3] or sumscat[4] or
sumscat[5]
    or sumscat1 or sumscat3 or sumscat5)
    begin
5      sumscatmax = (sumscat[0] > sumscat[2])? sumscat[0] : sumscat[2];
      sumscatmaxno0 = (sumscat[0] > sumscat[2])? sumscat1[3:0] : sumscat3[3:0];
      sumscatmaxno1 = (sumscatmax > sumscat[4])? sumscatmaxno0 : sumscat5[3:0];
    end
    assign mod12fftcnt = mod12(fftcnt);
10    assign sumscat1 = sumscat[1];
    assign sumscat3 = sumscat[3];
    assign sumscat5 = sumscat[5];

/*FOLDENDS*/
15 /*FOLDENDS*/
/*FOLDBEGINS 0 0 "ram"*/
always @(posedge clk)
    ramaddr_ <= ramaddr;
    always @(ramwritestop or valid or finishedsearch or fftcnt or carrier_number or
20 ramwritestop or ramaddr_ or fftdata)
    begin

        ramaddr = ramaddr_;
        if(!ramwritestop)
25         begin
            if(valid[0]&&!finishedsearch)
                ramaddr = {fftcnt[0],fftcnt[1],fftcnt[2],fftcnt[3],fftcnt[4],fftcnt[
                    5],fftcnt[6],
30                 fftcnt[7],fftcnt[8],fftcnt[9],fftcnt[10]};
            end
            else
                ramaddr = carrier_number;
                ramin = fftdata;
                wrstrb = !(ramwritestop&&valid[1]);
35     end
/*FOLDENDS*/

/*FOLDBEGINS 0 0 "modulus approximation function"*/
function [11:0] modulus;
40 input [11:0] i;
    input [11:0] j;
    reg [11:0] modi;
    reg [11:0] modj;
    begin
45     modi = (i[11]? ~i : i) + i[11];
        modj = (j[11]? ~j : j) + j[11];
        modulus = modi + modj;
    end
endfunction
50 /*FOLDENDS*/
/*FOLDBEGINS 0 0 "mod12"*/
function [3:0] mod12;
    input [10:0] count;
    reg [14:0] onetwelfth;
55 reg [7:0] modulus12;
    parameter TWELFTH = 12'haab;

```

```

begin
    onetwelfth = {count[0],count[1],count[2],count[3],count[4],count[5],count [6],
    count[7],count[8],count[9],count[10]} * TWELFTH;
    modulus12 = {onetwelfth[14:9],1'b0} + onetwelfth[14:9] + 4'h8;  /*12
5    mod12  = modulus12[7:4];
end
/*FOLDENDS*/
endfunction
endmodule

```

10

Listing 20

```

// ScclId: @(#)bch_decode.v      1.2 8/22/97
/*FOLDBEGINS 0 0 "copyright"*/
15  /******
// Copyright (c) 1997 Pioneer Digital Design Centre Limited
//
// NAME: BCH_rtl.v
//
20  // PURPOSE: BCH decoder for TPS pilots. Flags up to two error
// positions using search technique.
//
//*****
/*FOLDENDS*/
25  `define DATA0_SIZE 7'b0110100
    `define DATA1_SIZE 7'b0110111

module bch_decode (clk, resync, in_data, in_valid, in_finalwrite, out_valid, out_data);
/*FOLDBEGINS 0 0 "I/Os"*/
30  input clk, resync;
    input in_data, in_valid, in_finalwrite;
    output out_valid;
    output out_data;
    reg out_data;
35  reg out_valid;
/*FOLDENDS*/
/*FOLDBEGINS 0 0 "variables"*/
    reg resynch;
    reg valid;
40  reg finalwrite;
    reg indata;
    reg [6:0] S0;
    reg [6:0] S1;
    reg [6:0] S2;
45  reg [6:0] count;

    reg search1error, found2error, oneerror, twoerror;
    wire twoerror_;
    reg noerrors;
50  reg delay0, delay1, delay2;
    reg [6:0] Gs0;
    reg [6:0] Gs1;
    reg [6:0] Gs2;
/*FOLDENDS*/
55  always @(posedge clk)
    begin

```

```

/*FOLDBEGINS 0 2 "read in data and calculate syndromes"*/
resynch <= resynch;
if(resynch)
begin
5   valid <= 1'b0;
    S0 <= 7'b0;
    S1 <= 7'b0;
    S2 <= 7'b0;
end
10  else
begin
    valid <= in_valid;
    if(delay1&&twoerror_)
begin
15  /*FOLDBEGINS 0 4 "update after one in two errors found"*/
    S0 <= S0^Gs0;
    S1 <= S1^Gs1;
    S2 <= S2^Gs2;
    /*FOLDENDS*/
20  end
    else if(valid)
begin
    S0 <= indata ^ MULTA1(S0);
    S1 <= indata ^ MULTA2(S1);
25  S2 <= indata ^ MULTA3(S2);
end
end
indata <= in_data;
/*FOLDENDS*/
30  /*FOLDBEGINS 0 2 "out_valid control"*/
if(resynch)

begin
    delay0 <= 1'b0;
35  delay1 <= 1'b0;
    delay2 <= 1'b0;
    out_valid <= 1'b0;
    finalwrite <= 1'b0;
end
40  else
begin
    finalwrite <= in_finalwrite;
    if(valid&&finalwrite)
        delay0 <= 1'b1;
45  else
        if(count == `DATA1_SIZE-4)
            delay0 <= 1'b0;
        delay1 <= delay0;
        delay2 <= delay1;
50  out_valid <= delay2;
end
/*FOLDENDS*/
/*FOLDBEGINS 0 2 "error search algorithm"*/
if(delay0&&!delay1)
55  begin
        noerrors <= (S0 == 7'b0);

```

```

search1error <= (GFULL(S0,S1) == S2);
found2error <= 1'b0;
twoerror <= 1'b0;
count <= 7'b0;
5   Gs0 <= 7'h50;
    Gs1 <= 7'h20;
    Gs2 <= 7'h3d;
end
else
10  if(delay1)
    begin
        oneerror <= ((S0^Gs0) == 7'b0)&&search1error;
        twoerror_ <= twoerror_;
        if(twoerror_)
15         begin
            search1error <= 1'b1;
            found2error <= 1'b1;
        end
        Gs0 <= DIV1(Gs0);
        Gs1 <= DIV2(Gs1);
20         Gs2 <= DIV3(Gs2);
        count <= count + 1'b1;
    end
    out_data <= (twoerror_|oneerror)&&!noerrors;
25    /*FOLDENDS*/
    end
    assign twoerror_ = ( GFULL((S0^Gs0),(S1^Gs1)) ==
(S2^Gs2))&&!found2error&&!twoerror;
    /*FOLDBEGINS 0 0 "functions"*/
30    /*FOLDBEGINS 0 0 "GFULL function"*/

    function [6:0] GFULL;
    input [6:0] X;
    input [6:0] Y;
35    reg [6:0] A0, A1, A2, A3, A4, A5, A6;
    integer i;
    begin
        A0 = X;
        A1 = {A0[5],A0[4],A0[3],A0[2] ^ A0[6],A0[1],A0[0],A0[6]};
40        A2 = {A1[5],A1[4],A1[3],A1[2] ^ A1[6],A1[1],A1[0],A1[6]};
        A3 = {A2[5],A2[4],A2[3],A2[2] ^ A2[6],A2[1],A2[0],A2[6]};
        A4 = {A3[5],A3[4],A3[3],A3[2] ^ A3[6],A3[1],A3[0],A3[6]};
        A5 = {A4[5],A4[4],A4[3],A4[2] ^ A4[6],A4[1],A4[0],A4[6]};
        A6 = {A5[5],A5[4],A5[3],A5[2] ^ A5[6],A5[1],A5[0],A5[6]};
45
        for(i=0;i<7;i=i+1)
            begin
                A0[i] = A0[i] && Y[0];
                A1[i] = A1[i] && Y[1];
50                A2[i] = A2[i] && Y[2];
                A3[i] = A3[i] && Y[3];
                A4[i] = A4[i] && Y[4];
                A5[i] = A5[i] && Y[5];
                A6[i] = A6[i] && Y[6];
55            end
        GFULL = A0 ^ A1 ^ A2 ^ A3 ^ A4 ^ A5 ^ A6;

```

```

    end
    endfunction
    /*FOLDENDS*/
    /*FOLDBEGINS 0 0 "MULTA1 function"*/
5    function [6:0] MULTA1;
    input [6:0] X;
    begin
        MULTA1 = {X[5],X[4],X[3],X[2] ^ X[6],X[1],X[0],X[6]};
    end
10    endfunction
    /*FOLDENDS*/
    /*FOLDBEGINS 0 0 "MULTA2 function"*/
    function [6:0] MULTA2;
    input [6:0] X;
15    begin
        MULTA2 = {X[4],X[3],X[2]^X[6],X[1]^X[5],X[0],X[6],X[5]};
    end
    endfunction
    /*FOLDENDS*/
20    /*FOLDBEGINS 0 0 "MULTA3 function"*/
    function [6:0] MULTA3;
    input [6:0] X;
    begin
        MULTA3 = {X[3],X[2]^X[6],X[1]^X[5],X[0]^X[4],X[6],X[5],X[4]};
25    end
    endfunction
    /*FOLDENDS*/
    /*FOLDBEGINS 0 0 "DIV1 function"*/
    function [6:0] DIV1;
30    input [6:0] X;
    begin
        DIV1 = {X[0],X[6],X[5],X[4],X[3]^X[0],X[2],X[1]};
    end
    endfunction
35    /*FOLDENDS*/
    /*FOLDBEGINS 0 0 "DIV2 function"*/
    function [6:0] DIV2;
    input [6:0] X;
    begin
40    DIV2 = {X[1],X[0],X[6],X[5],X[4]^X[1],X[3]^X[0],X[2]};
    end
    endfunction
    /*FOLDENDS*/
    /*FOLDBEGINS 0 0 "DIV3 function"*/
45    function [6:0] DIV3;
    input [6:0] X;
    begin
        DIV3 = {X[2],X[1],X[0],X[6],X[5]^X[2],X[4]^X[1],X[3]^X[0]};
    end
50    endfunction
    /*FOLDENDS*/
    /*FOLDENDS*/
    /*FOLDBEGINS 0 0 ""*/
    //always @(posedge clk)
55    // $display(in_valid,,in_data,,in_finalwrite,,out_valid,,out_data,,S0,,S1,,S2 ...);
    //always @(psedge clk)

```



```

// $display(resynch,,in_valid,,in_data,,out_valid,,S0,,S1,,,count,,,delay0,,del
ay1,,delay2,,,
// ...,delay2,,noerrors,,oneerror,,twoerror,,out_data,,out_valid);
//always @(posedge clk)
5 // $display(in_valid,,in_data,,,out_valid,,out_data,,,S0,,S1,,S2,,,);
//always @(posedge clk)
// $display(in_valid,,in_data,,,out_valid,,out_data,,,S0,,S1,,S2,,,);
/*FOLDENDS*/
endmodule
10

```

Listing 21

```

// Scsid: @(#)tps.v      1.2 9/15/97
/*FOLDBEGINS 0 0 "copyright"*/
15 //*****
// Copyright (c) 1997 Pioneer Digital Design Centre Limited
//
// NAME tps_rtl.v
//
20 // PURPOSE: Demodulates TPS pilots using DPSK. Finds sync bits.
//   Corrects up to two errors using BCH.
//   (DPSK produces two errors for each transmission error)
// HISTORY:
// 15/9/97 PK Added scan IO ports, te, tdin, tdout
25 //
//*****
/*FOLDENDS*/
`define SYNCSEQ0 16'b0111011110101100
`define SYNCSEQ1 16'b1000100001010011
30 module tps (resync, clk, tps_valid, tps_pilot, tps_sync, tps_data, upsel, upaddr,
uprstr, lupdata,
te, tdin, tdout);
/*FOLDBEGINS 0 0 "i/os"*/
35 input resync, clk, tps_valid, tps_pilot, upsel, uprstr, te, tdin;
input [1:0] upaddr;
inout [7:0] lupdata;
output tps_sync, tdout;
output [30:0] tps_data;
/*FOLDENDS*/
40 /*FOLDBEGINS 0 0 "registers"*/
reg resynch;
reg [1:0] foundsync;
reg [66:0] tpsreg;
reg [15:0] syncreg;
45 reg [1:0] tpsvalid;
reg [1:0] pilot;
reg tps_sync;
reg [7:0] bch_count;
reg [2:0] bch_go;
50 reg bch_finalwrite;
wire bch_data;
wire bch_valid;
wire bch_error;
integer i;
55 wire upsel0;
wire upsel1;

```

```

        wire upsel2;
        wire upsel3;
        /*FOLDENDS*/

5    always @(posedge clk)
    begin
        /*FOLDBEGINS 0 2 "Synchronise to TPS"*/
        resynch <= resynch;
        if(tpsvalid[0]&&!(foundsync[0] || foundsync[1] || tps_sync))
10    begin
            tpsreg[66] <= pilot[1]^pilot[0];
            for(i=0;i<66;i=i+1)
                tpsreg[i] <= tpsreg[i+1];

15    end
        else
            if(bch_valid&&bch_error)
                tpsreg[bch_count] <= !tpsreg[bch_count];
            if(tpsvalid[0]&&(foundsync[0] || foundsync[1]))
20    begin
                syncreg[15] <= pilot[1]^pilot[0];
                for(i=0;i<15;i=i+1)
                    syncreg[i] <= syncreg[i+1];
            end

25    pilot[0] <= tps_pilot;
        pilot[1] <= pilot[0];
        if(resynch)
            begin
30    tpsvalid    <= 2'b0;
                tps_sync    <= 1'b0;
                bch_go      <= 3'b0;
                bch_finalwrite <= 1'b0;
                bch_count    <= 8'b0;
35    foundsync   <= 2'b0;
            end
        else
            begin
40    tpsvalid[0] <= tps_valid;
                tpsvalid[1] <= tpsvalid[0];
                bch_go[1] <= bch_go[0];
                bch_go[2] <= bch_go[1];
                bch_finalwrite <= (bch_count == 65)&&bch_go[2];
                if((bch_count == 52)&&bch_valid)
45    tps_sync <= 1'b1;
                /*FOLDBEGINS 0 2 "counter"*/
                if(bch_count == 66)
                    bch_count <= 8'b0;
                else if(tpsvalid[1]&&!(foundsync[0] || foundsync[1]))
50    begin
                    if(tpsreg[15:0] == `SYNCSEQ1)
                        bch_count <= 8'hfe;    //-2
                    if(tpsreg[15:0] == `SYNCSEQ0)
                        bch_count <= 8'hfe;    //-2
55    end
                else if(tpsvalid[1]&&(bch_count==15)&&(foundsync[0] || foundsync[1]))

```

```

        bch_count <= 8'hfe;    //-2
        else
        begin
            if(bch_valid || bch_go[0] || ((foundsync[0] || foundsync[1])&&tpsvalid[0]))
5         bch_count <= bch_count + 1'b1;
        end
        /*FOLDENDS*/
        /*FOLDBEGINS 0 2 "BCH + second SYNC reg control"*/
        if(bch_count == 66)
10         begin
            bch_go <= 3'b0;

            end
            else if(tpsvalid[1])
15         begin
            if(foundsync[0] || foundsync[1])
            begin
                if(bch_count==15)
                begin
20                 if(((syncreg[15:0] == `SYNCSEQ0)&&foundsync[1]) || ((syncreg[15:0]
                    == `SYNCSEQ1)&&foundsync[0]) )
                    bch_go[0] <= 1'b1;
                    else
                    foundsync <= 2'b0;
25                 end
                end
            else
            begin
                if(tpsreg[15:0] == `SYNCSEQ1)
30                 foundsync[1] <= 1'b1;
                if(tpsreg[15:0] == `SYNCSEQ0)
                foundsync[0] <= 1'b1;
            end
            end
35         /*FOLDENDS*/
        end
        /*FOLDENDS*/
        end
        assign bch_data = tpsreg[bch_count];
40         /*FOLDBEGINS 0 0 ""*/
        //always @(posedge clk)
        //begin
        // $write(tps_valid,,tps_sync,,tps_pilot,,tpsvalid[1],,pilot,,,,,
        // bch_finalwrite,,,,,bch_go[2],,bch_data,,bch_valid,,bch_error,,bch_count,,tps
45         _sync,,,,);
        // $displayb(tpsreg,,syncreg,,foundsync);
        //end
        /*FOLDENDS*/
        /*FOLDBEGINS 0 0 "micro access"*/
50         assign upsel0 = upsel&&uprstr&&!upaddr[1]&&!upaddr[0];
        assign upsel1 = upsel&&uprstr&&!upaddr[1]&& upaddr[0];
        assign upsel2 = upsel&&uprstr&& upaddr[1]&&!upaddr[0];
        assign upsel3 = upsel&&uprstr&& upaddr[1]&& upaddr[0];
        assign lupdata = upsel0? {1'b0,tps_data[30:24]} : 8'bz,
55         lupdata = upsel1? tps_data[23:16] : 8'bz,
        lupdata = upsel2? tps_data[15:8] : 8'bz,

```

```

        lupdata = upsel3? tps_data[7:0] : 8'bz;
/*FOLDENDS*/
assign tps_data = tpsreg[52:22];
bch_decode bch1 (.clk(clk), .resync(resync), .in_valid(bch_go[2]),
5  .in_finalwrite(bch_finalwrite), .in_data(bch_data),
        .out_valid(bch_valid), .out_data(bch_error));
endmodule

```

10

Listing 22

```
//SccsID = %W% %G%
```

15

```
//FOLDBEGINS 0 0 "Copyright (c) 1997 Pioneer Digital Design Centre Limited ..."
```

```
/*-----
   Copyright (c) 1997 Pioneer Digital Design Centre Limited

```

```
NAME: syndint_rtl.v
```

```
20 PURPOSE: <a one line description>
```

```
CREATED: Thu 14 Aug 1997 BY: Paul(Paul McCloy)
```

```
MODIFICATION HISTORY:
```

```
25 15/9/97 PK Increased width to 13 to allow for bad_carrier flag
```

```
-----*/
//FOLDENDS
```

```
30 //FOLDBEGINS 0 0 "module syndint ... <- top level"
```

```
////////////////////////////////////
```

```
module syndint
```

```
//FOLDBEGINS 0 0 "pins ..."
```

```
(
```

```
35     out_data,
        valid,
        d_symbol,
```

```
40     valid_in,
        demap_data,
        odd_symbol,
        symbol,
        carrier0,
        constellation,
```

```
45 //FOLDBEGINS 0 3 "ram pins ..."
    ram_a,
    ram_di,
```

```
50     ram_do,
        ram_wreq,
//FOLDENDS
```

```
//FOLDBEGINS 0 3 "scan pins ..."
```

```
55     tdin,
        tdout,
```

```
te,
//FOLDENDS

5      nrst,
      clk
);
//FOLDENDS

10      parameter WIDTH = 13; // Modified by PK 15/9/97; 12->13
      parameter ADDR_WIDTH = 11;

      //FOLDBEGINS 0 2 "outputs ..."
      output tdout;

15      output valid;
      output [17:0]out_data;
      output d_symbol;

      output [ADDR_WIDTH-1:0]ram_a;
20      output [WIDTH-1:0]ram_di;
      output ram_wreq;
      //FOLDENDS
      //FOLDBEGINS 0 2 "inputs ..."

25      input valid_in;
      input [WIDTH-1:0]demap_data;
      input odd_symbol;
      input symbol;
      input carrier0;
30      input [WIDTH-1:0]ram_do;
      input [1:0]constellation;

      input tdin, te;

35      input nrst, clk;
      //FOLDENDS
      //FOLDBEGINS 0 2 "regs / wires ..."

      //FOLDBEGINS 0 0 "inputs regs ..."
40      reg valid_in_reg;
      reg [WIDTH-1:0]demap_data_reg;
      reg odd_symbol_reg;
      reg symbol_reg;
45      reg [WIDTH-1:0]ram_do_reg;
      reg [1:0]constellation_reg;
      //FOLDENDS
      //FOLDBEGINS 0 0 "output regs ..."

50      reg valid;
      reg [17:0]out_data;
      reg d_symbol;

      reg [ADDR_WIDTH-1:0]ram_a;
55      reg [WIDTH-1:0]ram_di;
```

228

```

reg ram_wreq;
//FOLDENDS

//FOLDBEGINS 0 0 "instate_reg ..."
5
parameter INSTATE_WAIT_SYMBOL = 2'd0;
parameter INSTATE_WAIT_VALID = 2'd1;
parameter INSTATE_WRITE = 2'd2;
parameter INSTATE_WRITE_RAM = 2'd3;
10
reg [1:0]instate_reg;
//FOLDENDS
//FOLDBEGINS 0 0 "outstate_reg ..."

15
parameter OUTSTATE_WAIT_WRITEFINISHED = 3'd0;
parameter OUTSTATE_WAIT0 = 3'd1;
parameter OUTSTATE_WAIT1 = 3'd2;
parameter OUTSTATE_READRAM = 3'd3;
parameter OUTSTATE_WAIT2 = 3'd4;
20
parameter OUTSTATE_OUTPUTDATA = 3'd5;
parameter OUTSTATE_WAIT3 = 3'd6;

reg [2:0]outstate_reg;
//FOLDENDS
25

reg [ADDR_WIDTH-1:0]read_addr_reg;
reg [WIDTH-1:0]data_reg;
reg next_read_reg, next_write_reg;
reg frist_data_reg;
30
reg odd_read_reg, odd_write_reg;
reg sym_rst_read_reg, sym_rst_write_reg;

reg [17:0] demapped;
reg [3:0] iminus;
35
reg [3:0] qminus;
reg [8:0] outi;
reg [8:0] outq;
reg [5:0] demap;

40
//FOLDBEGINS 0 0 "wires ..."
wire [ADDR_WIDTH-1:0]address_read, address_write;
wire finished_read, finished_write;
wire valid_read, write_valid;
45

wire [5:0]ini, inq;
//FOLDENDS
//FOLDENDS

50
ag #(ADDR_WIDTH) r
//FOLDBEGINS 0 2 "pins ..."
(
.address(address_read),

55
.finished(finished_read),
.next(next_read_reg),

```

```

.random(odd_read_reg),
.sym_rst(sym_rst_read_reg),
.nrst(nrst),
.clk(clk)
5      );
      //FOLDENDS

      ag #(ADDR_WIDTH) w
      //FOLDBEGINS 0 2 "pins ..."
10     (
      .address(address_write),
      .finished(finished_write),
      .next(next_write_reg),
      .random(~odd_write_reg),
15     .sym_rst(sym_rst_write_reg),
      .nrst(nrst),
      .clk(clk)
      );
      //FOLDENDS
20
      //FOLDBEGINS 0 2 "latch inputs ..."
      always @(posedge clk)
      begin
25          valid_in_reg  <= valid_in;
          demap_data_reg <= demap_data;
          odd_symbol_reg <= odd_symbol;
          symbol_reg     <= symbol;
          ram_do_reg     <= ram_do;
          constellation_reg <= constellation;
30      end
      //FOLDENDS

      always @(posedge clk)
      begin
35          if( ~nrst )
              //FOLDBEGINS 0 4 "reset ..."
              begin
                  instate_reg <= INSTATE_WAIT_SYMBOL;
                  outstate_reg <= OUTSTATE_WAIT_WRITEFINISHED;
40          next_read_reg <= 0;
              end
              //FOLDENDS
              else
                  begin
45          //FOLDBEGINS 0 4 "input state machine ..."
                  //$write("DB(%0d %m): instate_reg=%0d   fw=%b\n",
                  //      $time, instate_reg, finished_write);
                  case (instate_reg)
                      INSTATE_WAIT_SYMBOL: begin
50                          sym_rst_write_reg <= 1;
                          next_write_reg <= 0;
                          ram_wreq <= 0;

                          if( symbol_reg )
55                          begin

```

```

//$write("DB(%0d %m): GOT = %x (NEW SYMBOL)\n", $time,
    demap_data_reg);
$write("DB(%0d %m): START WRITE\n", $time);
5      odd_write_reg <= odd_symbol_reg;
      data_reg <= demap_data_reg;
      instate_reg <= INSTATE_WRITE;
      end
      end
10     INSTATE_WAIT_VALID: begin
      ram_wreq <= 0;
      next_write_reg <= 0;
      if( finished_write )
      begin
15          $write("DB(%0d %m): END(1) WRITE\n", $time);
          instate_reg <= INSTATE_WAIT_SYMBOL;
          end
          else
          begin
20              if( valid_in_reg )
              begin
                  data_reg <= demap_data_reg;
                  instate_reg <= INSTATE_WRITE;
                  end
                  end
25          end
          INSTATE_WRITE: begin
              sym_rst_write_reg <= 0;
              next_write_reg <= 1;
              ram_a <= address_write;
30              //$write("DB(%0d %m): RWrite[%x] = %x\n", $time, address_write,
              data_reg);
              ram_di <= data_reg;
              ram_wreq <= 1;
              if( finished_write )
35              begin
                  $write("DB(%0d %m): END(2) WRITE\n", $time);
                  instate_reg <= INSTATE_WAIT_SYMBOL;
                  ram_wreq <= 0;
                  end
40              else
                  instate_reg <= INSTATE_WAIT_VALID;
                  end
      endcase
      //FOLDENDS
45  //FOLDBEGINS 0 4 "output state machine ..."
      //$write("DB(%0d %m): outstate_reg=%0d  nr:%b r:%b\n",
      //  $time, outstate_reg, next_read_reg, odd_symbol_reg);
      case (outstate_reg)
      OUTSTATE_WAIT_WRITEFINISHED: begin
50          sym_rst_read_reg <= 1;
          frist_data_reg <= 1;
          valid <= 0;

          if( finished_write )
55          begin
              odd_read_reg <= odd_write_reg;

```



```

        outstate_reg <= OUTSTATE_WAIT0;
        $write("DB(%0d %m): START READ\n", $time);
        //$write("DB(%0d %m): Read (NEW SYMBOL)\n", $time,
5         address_read);
        end
        end
OUTSTATE_WAIT0: begin
    sym_rst_read_reg <= 0;
    outstate_reg <= OUTSTATE_WAIT1;
10    end
OUTSTATE_WAIT1: begin
    outstate_reg <= OUTSTATE_READRAM;
    end
    OUTSTATE_READRAM: begin
15        //$write("DB(%0d %m): Read [%x]\n", $time, address_read);
        ram_a <= address_read;
        ram_wreq <= 0;
        next_read_reg <= 1;
        outstate_reg <= OUTSTATE_WAIT2;
20    end
OUTSTATE_WAIT2: begin
    next_read_reg <= 0;
    outstate_reg <= OUTSTATE_OUTPUTDATA;
    end
25    OUTSTATE_OUTPUTDATA: begin
        out_data <= {outi[8:6], outq[8:6], outi[5:3],
        outq[5:3], outi[2:0], outq[2:0]};
        valid <= 1;
        d_symbol <= frist_data_reg;
30        frist_data_reg <= 0;
        outstate_reg <= OUTSTATE_WAIT3;
    end
    OUTSTATE_WAIT3: begin
        valid <= 0;
35        if( finished_read )
            begin
                outstate_reg <= OUTSTATE_WAIT_WRITEFINISHED;
                $write("DB(%0d %m): END READ\n", $time);
            end
40        else
            outstate_reg <= OUTSTATE_WAIT0;
        end
    endcase
    //FOLDENDS
45    end
end

always @(constellation_reg or ini or inq)
//FOLDBEGINS 0 2 "demapper ..."
50    begin
        //FOLDBEGINS 0 2 "coarse demapping"

        iminus = {ini[5:3], 1'b0} -2'd3;
        qminus = {inq[5:3], 1'b0} -2'd3;
55        if(constellation_reg==2'b01)
            begin

```

```

    demap = { 2'b0,
              iminus[2],
              qminus[2],
              !(iminus[2]^iminus[1]),
5             !(qminus[2]^qminus[1])
              };
              //$writeb(demap,,);
              //$display(iminus,,ini[5:3]);
    end
10   else if(constellation_reg==2'b10)
    begin
        iminus = {ini[5:3],1'b0} -3'd7;
        qminus = {inq[5:3],1'b0} -3'd7;
        demap = { iminus[3],
15                qminus[3],
                !(iminus[3]^iminus[2]),
                !(qminus[3]^qminus[2]),
                (iminus[2]^iminus[1]),
                (qminus[2]^qminus[1])
20                };
    end
    else
        demap = 6'b0;

25   //FOLDENDS

    if(constellation_reg==2'b01)
    begin
        //FOLDBEGINS 0 4 "16QAM"
30       if(!iminus[1]&&iminus[0])
        begin
            outi[8:6] = 3'b0;
            outi[5:3] = demap[3]? 3'b111 : 3'b0;
            outi[2:0] = iminus[2]? ini[2:0] : ~ini[2:0];
35         end
        else
        begin
            outi[8:6] = 3'b0;
            outi[5:3] = ~ini[2:0];
40             outi[2:0] = 3'b111;
        end
        if(!qminus[1]&&qminus[0])
        begin
            outq[8:6] = 3'b0;
45             outq[5:3] = demap[2]? 3'b111 : 3'b0;
            outq[2:0] = qminus[2]? inq[2:0] : ~inq[2:0];
        end
        else
        begin
50             outq[8:6] = 3'b0;

            outq[5:3] = ~inq[2:0];
            outq[2:0] = 3'b111;
        end
55       end

        //FOLDENDS

```

```

        end
        else if(constellation_reg==2'b10)
        begin
//FOLDBEGINS 0 4 "64QAM"
5      if(!liminus[1])
        begin
            outi[8:6] = demap[5]? 3'b111 : 3'b0;
            outi[5:3] = demap[3]? 3'b111 : 3'b0;
            outi[2:0] = iminus[2]? ~ini[2:0] : ini[2:0];
10      end
        else if(!liminus[2])
        begin
            outi[8:6] = demap[5]? 3'b111 : 3'b0;
            outi[5:3] = iminus[3]? ini[2:0] : ~ini[2:0];
15      outi[2:0] = demap[1]? 3'b111 : 3'b0;
        end
        else
        begin
            outi[8:6] = ~ini[2:0];
20      outi[5:3] = demap[3]? 3'b111 : 3'b0;
            outi[2:0] = demap[1]? 3'b111 : 3'b0;
        end
        if(!lqminus[1])
        begin
25      outq[8:6] = demap[4]? 3'b111 : 3'b0;
            outq[5:3] = demap[2]? 3'b111 : 3'b0;
            outq[2:0] = qminus[2]? ~inq[2:0] : inq[2:0];
        end
        else if(!lqminus[2])
30      begin
            outq[8:6] = demap[4]? 3'b111 : 3'b0;
            outq[5:3] = qminus[3]? inq[2:0] : ~inq[2:0];
            outq[2:0] = demap[0]? 3'b111 : 3'b0;
        end
        else
35      begin
            outq[8:6] = ~inq[2:0];
            outq[5:3] = demap[2]? 3'b111 : 3'b0;
            outq[2:0] = demap[0]? 3'b111 : 3'b0;
40      end
        //FOLDENDS
        end
        else
        begin
45      //FOLDBEGINS 0 4 "QPSK"
            outi = {6'b0,~ini[2:0]};
            outq = {6'b0,~inq[2:0]};
            //FOLDENDS
            end
50      end
        //FOLDENDS

55      assign ini = ram_do_reg[11:6];
        assign inq = ram_do_reg[5:0];

```

```

endmodule
//FOLDENDS

//FOLDBEGINS 0 0 "module ag (address generation)..."
5 //////////////////////////////////////////////////
module ag
//FOLDBEGINS 0 0 "pins ..."
(
10     address,
        finished,

        next,
        random,
        sym_rst,
15     nrst,
        clk
);
//FOLDENDS
20     parameter ADDR_WIDTH = 12;

//FOLDBEGINS 0 2 "outputs ..."
25     output [ADDR_WIDTH-1:0] address;
        output finished;
//FOLDENDS
//FOLDBEGINS 0 2 "inputs ..."
        input next;
        input random;
30     input sym_rst;
        input nrst, clk;
//FOLDENDS

//FOLDBEGINS 0 2 "regs ..."
35     integer i;

        reg finished;
        reg [9:0] prsr_reg;
        reg [11:0] count_reg;
40     wire address_valid;
//FOLDENDS

        always @(posedge clk)
45     begin
            if( ~nrst )
                begin
                    count_reg <= 0;

50     prsr_reg <= 10'd0;
                end
            else
                begin
                    if(sym_rst)
55     begin
                        finished <= 0;
                    end
                end
            end
        end

```

```

        count_reg <= 0;
    end
    else
    if( next | (!address_valid & random) )
5      begin
        //$write("DB(%0d %m): Next(r:%d)\n", $time, random);
        if( random )
//FOLDBEGINS 0 8 "do the random stuff ..."
        begin
10          if( !address_valid )
            begin
//FOLDBEGINS 0 4 "drive the prsr ..."
            if( count_reg == 11'd0 )
15              prsr_reg <= 10'd0;
            else
              if( count_reg == 11'd1 )
                prsr_reg <= 10'd1;
            else
              begin
20                for(i=0;i<9;i=i+1)
                  prsr_reg[i] <= prsr_reg[i+1];
                  prsr_reg[9] <= prsr_reg[0] ^ prsr_reg[3];
                end
              end
            end
          end
        end
25      //FOLDENDS
        count_reg <= count_reg + 1;
        //$write("DB(%0d %m): count=%0d Rand(Retry)\n", $time,
        count_reg);
30      end
    else
    begin
        if( count_reg == 11'd2047 )
        begin
35          //$write("DB(%0d %m): *** FINISHED Rand\n", $time);
          finished <= 1;
          count_reg <= 0;
          prsr_reg <= 10'd0;
        end
    end
40      else
    begin
//FOLDBEGINS 0 6 "drive the prsr ..."
        if( count_reg == 11'd0 )
45          prsr_reg <= 10'd0;
        else
          if( count_reg == 11'd1 )
            prsr_reg <= 10'd1;
          else
50          begin
            for(i=0;i<9;i=i+1)
              prsr_reg[i] <= prsr_reg[i+1];
              prsr_reg[9] <= prsr_reg[0] ^ prsr_reg[3];
            end
          end
55          //FOLDENDS
          count_reg <= count_reg + 1;
        end
    end
  end

```

```

        // $write("DB(%0d %m): count=%0d Rand\n", $time, count_reg);
        finished <= 0;
    end
end
5      end
      //FOLDENDS
    else
//FOLDBEGINS 0 8 "do the sequential stuff ..."
begin
10      if( count_reg != 11'd1511 )
        begin
            // $write("DB(%0d %m): count=%0d Sequ\n", $time, count_reg);
            count_reg <= count_reg + 1;
            finished <= 0;
15      end
        else
        begin
            // $write("DB(%0d %m): *** FINISHED Sequ\n", $time);
            finished <= 1;
            count_reg <= 0;
20      end
        end
        //FOLDENDS
    end
25  end
    end
    end

//FOLDBEGINS 0 2 "assign address ..."
assign address = (random) ? ({count_reg[0], // 10
30      prsr_reg[2], // 9
        prsr_reg[5], // 8
        prsr_reg[8], // 7
        prsr_reg[3], // 6
        prsr_reg[7], // 5
35      prsr_reg[0], // 4
        prsr_reg[1], // 3
        prsr_reg[4], // 2
        prsr_reg[6], // 1
        prsr_reg[9]}}): // 0
40      count_reg;
        //FOLDENDS

    assign address_valid = (address < 11'd1512);
    endmodule
45  //FOLDENDS

```

Listing 23

```

//SccsID: "@(#)bitdeint.v 1.4 9/14/97"
//FOLDBEGINS 0 0 "Copyright (c) 1997 Pioneer Digital Design Centre Limited"
50  /*****
    Copyright (c) 1997 Pioneer Digital Design Centre Limited

    NAME: bitdeint_rtl.v

55  PURPOSE: bit deinterleaver

```

CREATED: Wed 23 Jul 1997 BY: Paul(Paul McCloy)

MODIFICATION HISTORY:

```
5 *****/
//FOLDENDS

module bitdeint
//FOLDBEGINS 0 2 "pins ..."
10 (
    i_data,
    q_data,
    discard_i,
    discard_q,
15     valid, // output

    //FOLDBEGINS 0 2 "ram0 pins ..."

20     ram0_a,
    ram0_di,
    ram0_do,
    ram0_wreq,
    ram0_ce,
25     //FOLDENDS
    //FOLDBEGINS 0 2 "ram1 pins ..."

    ram1_a,
    ram1_di,
30     ram1_do,
    ram1_wreq,
    ram1_ce,
    //FOLDENDS
    //FOLDBEGINS 0 2 "ram2 pins ..."

35     ram2_a,
    ram2_di,
    ram2_do,
    ram2_wreq,
40     ram2_ce,
    //FOLDENDS

    bad_carrier,
    valid_in,
45     data_in,
    symbol,
    constellation, // constellation
    alpha, // does not do anything yet

50     //FOLDBEGINS 0 2 "scan pins ..."
    tdin,
    tdout,
    te,
    //FOLDENDS

55     nrst,
```

```

        clk
    );
    //FOLDENDS

5    parameter SBW = 3; // soft bit width

    //FOLDBEGINS 0 2 "outputs ..."
    //FOLDBEGINS 0 0 "ram0 outputs ..."
    output [6:0]ram0_a;
10    output [((SBW+1)<<1)-1:0]ram0_di;
    output ram0_ce;
    output ram0_wreq;
    //FOLDENDS
    //FOLDBEGINS 0 0 "ram1 outputs ..."
15    output [6:0]ram1_a;
    output [((SBW+1)<<1)-1:0]ram1_di;
    output ram1_ce;
    output ram1_wreq;
    //FOLDENDS
20    //FOLDBEGINS 0 0 "ram2 outputs ..."
    output [6:0]ram2_a;
    output [((SBW+1)<<1)-1:0]ram2_di;
    output ram2_ce;
    output ram2_wreq;
25    //FOLDENDS

    output tdout;

    output [SBW-1:0]i_data;
30    output [SBW-1:0]q_data;
    output discard_i;
    output discard_q;

    output valid;
35    //FOLDENDS
    //FOLDBEGINS 0 2 "inputs ..."

    input [((SBW+1)<<1)-1:0]ram0_do;
40    input [((SBW+1)<<1)-1:0]ram1_do;
    input [((SBW+1)<<1)-1:0]ram2_do;

    input bad_carrier;
    input valid_in;
45    input [((SBW<<2)+(SBW<<1))-1:0]data_in; // 6*SBW bits
    input symbol;
    input [1:0] constellation;
    input [2:0] alpha;

50    input tdin, te;

    input nrst, clk;
    //FOLDENDS

55    //FOLDBEGINS 0 2 "reg / wire ..."
    //FOLDBEGINS 0 0 "outputs ..."

```



```

//FOLDBEGINS 0 0 "ram0 regs ..."
reg [6:0]ram0_a;
reg [((SBW+1)<<1)-1:0]ram0_di;
reg ram0_ce;
5 reg ram0_wreq;
//FOLDENDS
//FOLDBEGINS 0 0 "ram1 regs ..."
reg [6:0]ram1_a;
reg [((SBW+1)<<1)-1:0]ram1_di;
10 reg ram1_ce;
reg ram1_wreq;
//FOLDENDS
//FOLDBEGINS 0 0 "ram2 regs ..."
reg [6:0]ram2_a;
15 reg [((SBW+1)<<1)-1:0]ram2_di;
reg ram2_ce;
reg ram2_wreq;
//FOLDENDS

20 reg [SBW-1:0]i_data;
reg [SBW-1:0]q_data;
reg discard_i;
reg discard_q;

25 reg valid;
//FOLDENDS
//FOLDBEGINS 0 0 "inputs ..."

reg valid_in_reg;
30 reg [((SBW<<2)+(SBW<<1))-1:0]data_in_reg; // 6*SBW bits
reg symbol_reg, bad_carrier_reg;

reg [1:0] constellation_reg;
reg [2:0] alpha_reg;
35 reg [((SBW+1)<<1)-1:0]ram0_do_reg;
reg [((SBW+1)<<1)-1:0]ram1_do_reg;
reg [((SBW+1)<<1)-1:0]ram2_do_reg;

//FOLDENDS

40 reg [6:0]i0_adr_reg;
reg [6:0]i1_adr_reg;
reg [6:0]i2_adr_reg;
reg [6:0]i3_adr_reg;
45 reg [6:0]i4_adr_reg;
reg [6:0]i5_adr_reg;

reg [2:0] mode_reg;
reg [(SBW<<2)+(SBW<<1)-1:0]data_reg; // 6*(SBW) bits
50 reg [((SBW+1)<<1)+SBW:0]i_out_buf_reg, q_out_buf_reg; // 3*(SBW+1) bits

reg ram_filled_reg, out_buf_full_reg, bad_car_reg;

wire [SBW:0] i0_in, q0_in, i1_in, q1_in, i2_in, q2_in;
55 wire [SBW:0] i0_ram, q0_ram, i1_ram, q1_ram, i2_ram, q2_ram;
//FOLDENDS

```

```

//FOLDBEGINS 0 2 "latch inputs ..."
always @(posedge clk)
begin
    bad_carrier_reg <= bad_carrier;
    valid_in_reg   <= valid_in;
    data_in_reg    <= data_in;
    symbol_reg     <= symbol;
    constellation_reg <= constellation;
    alpha_reg      <= alpha;
    ram0_do_reg    <= ram0_do;
    ram1_do_reg    <= ram1_do;
    ram2_do_reg    <= ram2_do;
end
//FOLDENDS

15
always @(posedge clk)
begin
    if( ~nrst )
    //FOLDBEGINS 0 4 "reset ..."
20
    begin
        mode_reg <= 2'b00;
        valid <= 0;
        i0_adr_reg <= 0;
        i1_adr_reg <= 63;
25
        i2_adr_reg <= 105;
        i3_adr_reg <= 42;
        i4_adr_reg <= 21;
        i5_adr_reg <= 84;

30
        i_out_buf_reg <= 0;
        q_out_buf_reg <= 0;
        ram_filled_reg <= 0;
        out_buf_full_reg <= 0;
35
    end
    //FOLDENDS
    else
    begin
        if( valid_in_reg )
40
        //FOLDBEGINS 0 6 "start cycle ...."
        begin
            data_reg <= data_in_reg;
            bad_car_reg <= bad_carrier_reg;
            // $write("DB(%0d %m): data_reg=%X(%b.%b.%b)\n", $time, data_in_reg,
45
            //   bad_carrier, bad_carrier_reg, bad_car_reg);
            //FOLDBEGINS 0 2 "logic to read i0,1,2 ..."
            ram0_a <= i0_adr_reg;
            ram0_wreq <= 0;

50
            ram1_a <= i1_adr_reg;
            ram1_wreq <= 0;

            ram2_a <= i2_adr_reg;
            ram2_wreq <= 0;
55
        end
        //FOLDENDS
    end
end

```

```

ram0_ce <= 1;
ram1_ce <= (constellation_reg == 2'b00) |
           (constellation_reg == 2'b01);
ram2_ce <= (constellation_reg == 2'b10);
5
//FOLDBEGINS 0 2 "output i1 and q1 ..."
if( out_buf_full_reg & (constellation_reg != 2'b00))
begin
10
    valid <= 1;

    i_data <= i_out_buf_reg[((SBW+1)<<1)-2:(SBW+1)];
    discard_i <= i_out_buf_reg[((SBW+1)<<1)-1];

    q_data <= q_out_buf_reg[((SBW+1)<<1)-2:(SBW+1)];
15
    discard_q <= q_out_buf_reg[((SBW+1)<<1)-1];

    //$write("DB(%0d %m): OUT(1):%x %x\n", $time,
    //      i_out_buf_reg[((SBW+1)<<1)-2:(SBW+1)],
    //      q_out_buf_reg[((SBW+1)<<1)-2:(SBW+1)]);
20
end
//FOLDENDS

mode_reg <= 3'b001;
end
25
//FOLDENDS
else
begin
//$write("DB(%0d %m): m=%b\n", $time, mode_reg);

30
case( mode_reg )
//FOLDBEGINS 0 8 "3'b001: ..."
3'b001: begin
//FOLDBEGINS 0 4 "logic to read q0,1,2 ..."
    ram0_a <= i3_adr_reg;
35
    ram0_wreq <= 0;

    ram1_a <= i4_adr_reg;
    ram1_wreq <= 0;

    ram2_a <= i5_adr_reg;
    ram2_wreq <= 0;
40
    //FOLDENDS
    valid <= 0;
    mode_reg <= 3'b010;
45
end
//FOLDENDS
//FOLDBEGINS 0 8 "3'b010: ..."
3'b010: begin
mode_reg <= 3'b011;
50
//FOLDBEGINS 0 4 "output i2 and q2 ..."
if( out_buf_full_reg & (constellation_reg == 2'b10))
begin
    valid <= 1;

55
    i_data <= i_out_buf_reg[SBW-1:0];
    discard_i <= i_out_buf_reg[SBW];

```

242

```

        q_data <= q_out_buf_reg[SBW-1:0];
        discard_q <= q_out_buf_reg[SBW];

        //$write("DB(%0d %m): OUT(2):%x %x\n", $time,
5         //      i_out_buf_reg[SBW-1:0],
        //      q_out_buf_reg[SBW-1:0]);
    end
    //FOLDENDS
end
10 //FOLDENDS
//FOLDBEGINS 0 8 "3'b011: ...      "
3'b011: begin
    valid <= 0;

15     //$write("DB(%0d %m): ram read i0:%x i1:%x i2:%x\n",
        //      $time,
        //      ram0_do_reg[((SBW+1)<<1)-1:SBW+1],
        //      ram1_do_reg[((SBW+1)<<1)-1:SBW+1],
20     //      ram2_do_reg[((SBW+1)<<1)-1:SBW+1]);

    i_out_buf_reg <= {ram0_do_reg[((SBW+1)<<1)-1:SBW+1],
        ram1_do_reg[((SBW+1)<<1)-1:SBW+1],
        ram2_do_reg[((SBW+1)<<1)-1:SBW+1]};

25 //FOLDBEGINS 0 4 "logic to write new i0,1,2 ..."

    ram0_a <= i0_adr_reg;
    ram0_wreq <= 1;
    ram0_di <= {i0_in, q0_ram};

30     ram1_a <= i1_adr_reg;
    ram1_wreq <= 1;
    ram1_di <= {i1_in, q1_ram};

    ram2_a <= i2_adr_reg;
35     ram2_wreq <= 1;
    ram2_di <= {i2_in, q2_ram};
    //FOLDENDS
    mode_reg <= 3'b100;
40     end
    //FOLDENDS
//FOLDBEGINS 0 8 "3'b100: ...      "
3'b100: begin

45     //$write("DB(%0d %m): ram read q0:%x q1:%x q2:%x\n",
        //      $time,
        //      ram0_do_reg[SBW:0],
        //      ram1_do_reg[SBW:0],
50     //      ram2_do_reg[SBW:0]);

    q_out_buf_reg <= {ram0_do_reg[SBW:0],
        ram1_do_reg[SBW:0],
        ram2_do_reg[SBW:0]};

55     out_buf_full_reg <= ram_filled_reg;
    //FOLDBEGINS 0 4 "logic to write new q0,1,2 ..."

```

3

```

ram0_a <= i3_adr_reg;
ram0_wreq <= 1;
ram0_di <= {i0_ram, q0_in};

5      ram1_a <= i4_adr_reg;
      ram1_wreq <= 1;
      ram1_di <= {i1_ram, q1_in};

      ram2_a <= i5_adr_reg;
10     ram2_wreq <= 1;
      ram2_di <= {i2_ram, q2_in};
      //FOLDENDS

//FOLDBEGINS 0 4 "output i0 and q0 ..."
15     if( out_buf_full_reg )
      begin
          valid <= 1;
          i_data <= i_out_buf_reg[(((SBW+1)<<1)+SBW-1:((SBW+1)<<1))];
          discard_i <= i_out_buf_reg[(((SBW+1)<<1)+SBW)];
20         q_data <= q_out_buf_reg[(((SBW+1)<<1)+SBW-1:((SBW+1)<<1))];
          discard_q <= q_out_buf_reg[(((SBW+1)<<1)+SBW)];

          //$write("DB(%0d %m): OUT(0):%x %x\n", $time,
25         //    i_out_buf_reg[(((SBW+1)<<1)+SBW-1:((SBW+1)<<1))],
          //    q_out_buf_reg[(((SBW+1)<<1)+SBW-1:((SBW+1)<<1))];
          end
          //FOLDENDS
30         mode_reg <= 3'b101;
          end
          //FOLDENDS
          //FOLDBEGINS 0 8 "3'b101: ..."
          3'b101:begin
35         valid <= 0;
          //FOLDBEGINS 0 4 "increment ram address ..."

          if( i0_adr_reg == 7'd125 )
          begin
40             i0_adr_reg <= 0;
            //FOLDBEGINS 0 2 "do i1_adr_reg (63 offset)..."
            i1_adr_reg <= (i1_adr_reg == 7'd20) ? 7'd84 :
            (i1_adr_reg == 7'd41) ? 7'd105 :
            (i1_adr_reg == 7'd62) ? 7'd0 :
45             (i1_adr_reg == 7'd83) ? 7'd21 :
            (i1_adr_reg == 7'd104) ? 7'd42 :
            7'd63 ;
            //FOLDENDS

            //FOLDBEGINS 0 2 "do i2_adr_reg (105 offset)..."
50             i2_adr_reg <= (i2_adr_reg == 7'd20) ? 7'd42 :
            (i2_adr_reg == 7'd41) ? 7'd63 :
            (i2_adr_reg == 7'd62) ? 7'd84 :
            (i2_adr_reg == 7'd83) ? 7'd105 :
            (i2_adr_reg == 7'd104) ? 7'd0 :
55             7'd21 ;
            //FOLDENDS

```

244

```

//FOLDBEGINS 0 2 "do i3_adr_reg (42 offset)..."
i3_adr_reg <= (i3_adr_reg == 7'd20) ? 7'd105 :
    (i3_adr_reg == 7'd41) ? 7'd0 :
    (i3_adr_reg == 7'd62) ? 7'd21 :
    (i3_adr_reg == 7'd83) ? 7'd42 :
    (i3_adr_reg == 7'd104) ? 7'd63 :
    7'd84 ;
//FOLDENDS

//FOLDBEGINS 0 2 "do i4_adr_reg (21 offset)..."
i4_adr_reg <= (i4_adr_reg == 7'd20) ? 7'd0 :
    (i4_adr_reg == 7'd41) ? 7'd21 :
    (i4_adr_reg == 7'd62) ? 7'd42 :
    (i4_adr_reg == 7'd83) ? 7'd63 :
    (i4_adr_reg == 7'd104) ? 7'd84 :
    7'd105 ;
//FOLDENDS

//FOLDBEGINS 0 2 "do i5_adr_reg (84 offset)..."
i5_adr_reg <= (i5_adr_reg == 7'd20) ? 7'd63 :
    (i5_adr_reg == 7'd41) ? 7'd84 :
    (i5_adr_reg == 7'd62) ? 7'd105 :
    (i5_adr_reg == 7'd83) ? 7'd0 :
    (i5_adr_reg == 7'd104) ? 7'd21 :
    7'd42 ;
//FOLDENDS

    ram_filled_reg <= 1;
    end
    else
    begin
    i0_adr_reg <= i0_adr_reg + 1;
    i1_adr_reg <= (i1_adr_reg == 7'd125) ? 0 : i1_adr_reg + 1;
    i2_adr_reg <= (i2_adr_reg == 7'd125) ? 0 : i2_adr_reg + 1;
    i3_adr_reg <= (i3_adr_reg == 7'd125) ? 0 : i3_adr_reg + 1;
    i4_adr_reg <= (i4_adr_reg == 7'd125) ? 0 : i4_adr_reg + 1;
    i5_adr_reg <= (i5_adr_reg == 7'd125) ? 0 : i5_adr_reg + 1;
    end
    //FOLDENDS
    end
    //FOLDENDS
    endcase
    end
    end

end

assign i0_in = { bad_car_reg,
data_reg[(SBW<<2)+(SBW<<1)-1 : (SBW<<2)+SBW]};
assign q0_in = { bad_car_reg,
data_reg[(SBW<<2)+SBW-1 : SBW<<2]};
assign i1_in = { bad_car_reg,
data_reg[(SBW<<2)-1 : (SBW<<1)+SBW]};
assign q1_in = { bad_car_reg,
data_reg[(SBW<<1)+SBW-1 : SBW<<1]};
assign i2_in = { bad_car_reg,
data_reg[(SBW<<1)-1 : SBW]};
assign q2_in = { bad_car_reg,

```

```

    data_reg[SBW-1 :0]];

    assign i0_ram = i_out_buf_reg[((SBW+1)<<1)+SBW:((SBW+1)<<1)];
    assign q0_ram = q_out_buf_reg[((SBW+1)<<1)+SBW:((SBW+1)<<1)];
5    assign i1_ram = i_out_buf_reg[((SBW+1)<<1)-1:SBW+1];
    assign q1_ram = q_out_buf_reg[((SBW+1)<<1)-1:SBW+1];
    assign i2_ram = i_out_buf_reg[SBW:0];
    assign q2_ram = q_out_buf_reg[SBW:0];

10    endmodule

```

Listing 24

```

// Sccsid: %W% %G%
/*****
15    Copyright (c) 1997 Pioneer Digital Design Centre Limited
    *****/

20    module acc_prod (clk, resync, load, symbol, new_phase, old_phase, xcount,
        acc_out);

        input clk, resync, load, symbol;
        input [10:0] xcount;
25    input [13:0] new_phase, old_phase;
        output [29:0] acc_out;

        reg [29:0] acc_out;
        reg [29:0] acc_int;
30    reg [14:0] diff;
        reg [25:0] xdiff;

        reg sign;
        reg [14:0] mod_diff;
35    reg [25:0] mod_xdiff;

        always @ (posedge clk)
40    begin
        if (resync)
        begin
            acc_out <= 0;
            acc_int <= 0;
45    end

        else
        begin
            if (load)
50    acc_int <= acc_int + {xdiff[25], xdiff[25], // sign extend
                xdiff[25], xdiff[25], xdiff};
            if (symbol)
            begin
55    acc_out <= acc_int;
            acc_int <= 0;
            end

```

```

end
end

always @ (new_phase or old_phase or xcount)
5: begin
    diff = {new_phase[13], new_phase} // sign extend up to allow
        - {old_phase[13], old_phase}; // differences up to 360
    sign = diff[14];
    mod_diff = sign ? (~diff + 1) : diff;
10: mod_xdiff = mod_diff * {4'b0, xcount};
    xdiff = sign ? (~mod_xdiff + 1) : mod_xdiff;
end

endmodule
15:

```

Listing 25

```

// Scsid: %W% %G%
/*****
20: Copyright (c) 1997 Pioneer Digital Design Centre Limited
*****/

25: module acc_simple (clk, resync, load, symbol, new_phase, old_phase, acc_out);

    input clk, resync, load, symbol;
    input [13:0] new_phase, old_phase;
30: output [20:0] acc_out;

    reg [20:0] acc_out;
    reg [20:0] acc_int;
    reg [14:0] diff;
35:

    always @ (posedge clk)
    begin
        if (resync)
40:         begin
            acc_out <= 0;
            acc_int <= 0;
        end

        else
45:         begin
            if (load)
                acc_int <= acc_int + {diff[14], diff[14], // sign extend
50:                    diff[14], diff[14],
                    diff[14], diff[14], diff};

            if (symbol)
                begin
                    acc_out <= acc_int;
                    acc_int <= 0;
55:                 end
            end
        end
    end

```



```

end

always @ (new_phase or old_phase)
    diff = {new_phase[13], new_phase} // sign extend up to allow
5      - {old_phase[13], old_phase}; // differences up to 360

always @ (diff or load)
begin: display

10    reg[14:0] real_diff;

    if (load)
    begin
        if (diff[14])
15        begin
            real_diff = (~diff + 1);
            $display ("diff = -%0d", real_diff);
        end
        else
20        $display ("diff = %0d", diff);
        end
    end // display

endmodule

25

```

Listing 26

```

// SccsId: %W% %G%
/*****
30    Copyright (c) 1997 Pioneer Digital Design Centre Limited
    *****/

35    module addr_gen (clk, resync, u_symbol, uc_pilot, got_phase, en, load, guard,
        addr, xcount, guard_reg, symbol);

        input clk, resync, u_symbol, uc_pilot, got_phase;
        input [1:0] guard;
40        output en, load, symbol;
        output [1:0] guard_reg;
        output [9:0] addr;
        output [10:0] xcount;

45        reg en, load, load_p, inc_count2, symbol;
        reg [1:0] guard_reg;
        reg [5:0] count45;
        reg [10:0] xcount;
        reg [9:0] addr;

50

        always @ (posedge clk)
        begin
            if (resync)
55            begin
                count45 <= 0;
            end
        end
    end

```

```
load_p <= 0;
load <= 0;
inc_count2 <= 0;
symbol <= 0;
5 guard_reg <= 0;
end

else
begin
10 if (u_symbol)
begin
inc_count2 <= 1;
guard_reg <= guard;
end
15 if (inc_count2 && uc_pilot)
begin
inc_count2 <= 0;
count45 <= 0;
end
20 if (got_phase)
count45 <= count45 + 1;
load_p <= en;
load <= load_p;
symbol <= (inc_count2 && uc_pilot);
25
addr <= count45;
en <= got_phase && !resync && (count45 < 45); // !! 45 ?
end
end
30
always @ (count45)
case (count45)
1: xcount = 1;
2: xcount = 49;
35 3: xcount = 55;
4: xcount = 88;
5: xcount = 142;
6: xcount = 157;
7: xcount = 193;
40 8: xcount = 202;
9: xcount = 256;
10: xcount = 280;
11: xcount = 283;
12: xcount = 334;
45 13: xcount = 433;
14: xcount = 451;
15: xcount = 484;
16: xcount = 526;
17: xcount = 532;
50 18: xcount = 619;
19: xcount = 637;
20: xcount = 715;
21: xcount = 760;
22: xcount = 766;
55 23: xcount = 781;
24: xcount = 805;
```

```

25: xcount = 874;
26: xcount = 889;
27: xcount = 919;
28: xcount = 940;
5  29: xcount = 943;
30: xcount = 970;
31: xcount = 985;
32: xcount = 1051;
33: xcount = 1102;
10 34: xcount = 1108;
35: xcount = 1111;
36: xcount = 1138;
37: xcount = 1141;
38: xcount = 1147;
15 39: xcount = 1207;
40: xcount = 1270;
41: xcount = 1324;
42: xcount = 1378;
43: xcount = 1492;
20 44: xcount = 1684;
45: xcount = 1705;
    default: xcount = 0;
    endcase
endmodule

```

25

Listing 27

```

// SccsId: %W% %G%
/*****
30  Copyright (c) 1997 Pioneer Digital Design Centre Limited
    *****/

35  module avg_8 (clk, resync, symbol, in_data, avg_out);

    parameter phase_width = 12;

    input clk, resync, symbol;
40  input [phase_width-2:0] in_data;
    output [phase_width-2:0] avg_out;

    reg [phase_width-2:0] avg_out;
    reg [phase_width-2:0] store [7:0];
45

    wire [phase_width-2:0] store7 = store[7];
    wire [phase_width-2:0] store6 = store[6];
    wire [phase_width-2:0] store5 = store[5];
50  wire [phase_width-2:0] store4 = store[4];
    wire [phase_width-2:0] store3 = store[3];
    wire [phase_width-2:0] store2 = store[2];
    wire [phase_width-2:0] store1 = store[1];
55  wire [phase_width-2:0] store0 = store[0];

```

```

wire [phase_width+1:0] sum = ({store7[phase_width-2], store7[phase_width-2],
store7[phase_width-2], store7}
+ {store6[phase_width-2], store6[phase_width-2], store6[phase_width-2],
store6}
5   + {store5[phase_width-2], store5[phase_width-2], store5[phase_width-2],
store5}
+ {store4[phase_width-2], store4[phase_width-2], store4[phase_width-2],
store4}
10  + {store3[phase_width-2], store3[phase_width-2], store3[phase_width-2],
store3}
+ {store2[phase_width-2], store2[phase_width-2], store2[phase_width-2],
store2}
+ {store1[phase_width-2], store1[phase_width-2], store1[phase_width-2],
store1}
15  + {store0[phase_width-2], store0[phase_width-2], store0[phase_width-2],
store0});

always @ (posedge clk)
begin
20  if (resync)
begin
store[7] <= 0;
store[6] <= 0;
store[5] <= 0;
25  store[4] <= 0;
store[3] <= 0;
store[2] <= 0;
store[1] <= 0;
store[0] <= 0;
30  avg_out <= 0;
end
else if (symbol)
begin
35  store[7] <= store[6];
store[6] <= store[5];
store[5] <= store[4];
store[4] <= store[3];
store[3] <= store[2];
store[2] <= store[1];
40  store[1] <= store[0];
store[0] <= in_data;
avg_out <= sum >> 3;
end
end
45  endmodule

```

Listing 28

```

// SccsId: %W% %G%
50  /*****
Copyright (c) 1997 Pioneer Digital Design Centre Limited
*****/

55  module twowire26 (clk, rst, in_valid, din, out_accept, out_valid, in_accept,

```

dout, set);

```

5  input clk, rst, set, in_valid, out_accept;
   input [25:0] din;
   output in_accept, out_valid;
   output [25:0] dout;
   reg in_accept, out_valid, acc_int, acc_int_reg, in_valid_reg, val_int;
10  reg [25:0] dout, din_reg;

   always @ (posedge clk)
   begin
       if (rst)
           out_valid <= 0;
15  else if (acc_int || set)
           out_valid <= val_int;

       if (in_accept)
       begin
20  in_valid_reg <= in_valid;
           din_reg <= din;
       end

       if (acc_int)
25  dout <= in_accept ? din : din_reg;

       if (set)
           acc_int_reg <= 1;
       else
30  acc_int_reg <= acc_int;
       end

       always @ (out_accept or out_valid or acc_int_reg or in_valid or in_valid_reg)
       begin
35  acc_int = out_accept || !out_valid;
           in_accept = acc_int_reg || !in_valid_reg;
           val_int = in_accept ? in_valid : in_valid_reg;
       end

40  endmodule

module buffer (clk, nrst, resync, u_symbol_in, uc_pilot_in, ui_data_in,
45  uq_data_in, u_symbol_out, uc_pilot_out, ui_data_out,
   uq_data_out, got_phase);

   input clk, nrst, resync, u_symbol_in, uc_pilot_in, got_phase;
   input [11:0] ui_data_in, uq_data_in;
   output u_symbol_out, uc_pilot_out;
50  output [11:0] ui_data_out, uq_data_out;

   reg u_symbol_out, uc_pilot_out, accept;
   wire u_symbol_o, uc_pilot_o;
   reg [11:0] ui_data_out, uq_data_out;
55  wire [11:0] ui_data_o, uq_data_o;
   wire a, v;

```

```

wire [25:0] d;

wire in_valid = u_symbol_in || uc_pilot_in;
wire rst = !nrst || resync;
5

twowire26 tw1 (.clk(clk), .rst(rst), .in_valid(in_valid), .din({u_symbol_in,
    uc_pilot_in, ui_data_in, uq_data_in}), .out_accept(a),
    .out_valid(v), .in_accept(), .dout(d), .set(1'b0));
10

twowire26 tw2 (.clk(clk), .rst(rst), .in_valid(v), .din(d),
    .out_accept(accept), .out_valid(out_valid), .in_accept(a),
    .dout({u_symbol_o, uc_pilot_o, ui_data_o, uq_data_o}),
    .set(1'b0));
15

always @ (u_symbol_o or uc_pilot_o or ui_data_o or uq_data_o or out_valid or
    accept)
begin
20   if (out_valid && accept)
    begin
        u_symbol_out = u_symbol_o;
        uc_pilot_out = uc_pilot_o;
        ui_data_out = ui_data_o;
25        uq_data_out = uq_data_o;
    end
    else
    begin
        u_symbol_out = 0;
30        uc_pilot_out = 0;
        ui_data_out = 0;
        uq_data_out = 0;
    end
    end
35

always @ (posedge clk)
begin
    if (rst || got_phase)
        accept <= 1;
40    else if (uc_pilot_out)
        accept <= 0;
    end

endmodule
45

```

Listing 29

```

// SccsId: %W% %G%
/*****
50   Copyright (c) 1997 Pioneer Digital Design Centre Limited
    *****/

55
module divide (clk, go, numer, denom, answ, got);

```

```

/*****
this divider is optimised on the principal that the answer will always be
less than 1 - ie denom > numer
*****/
5
input clk, go;
input [10:0] numer, denom;
output got;
output [10:0] answ;

10
reg got;
reg [10:0] answ;
reg [20:0] sub, internal;
reg [3:0] dcount;

15
always @ (posedge clk)
begin
20
if (go)
begin
dcount <= 0;
internal <= numer << 10;
sub <= denom << 9;
end
25
if (dcount < 11)
begin
if (internal > sub)
begin
30
internal <= internal - sub;
answ[10 - dcount] <= 1;
end
else
begin
35
internal <= internal;
answ[10 - dcount] <= 0;
end

sub <= sub >> 1;
dcount <= dcount + 1;
40
end

got <= (dcount == 10);
end

45
endmodule

Listing 30

// Sccsid: %W% %G%
/*****
50
Copyright (c) 1997 Pioneer Digital Design Centre Limited
*****/

55
module fserr_str (clk, nrst, resync, u_symbol, uc_pilot, ui_data, uq_data, guard,

```

```

    freq_sweep, sr_sweep, lupdata, upaddr, upwstr, uprstr, upsel1,
    upsel2, ram_di, te, tdin, freq_err, samp_err, ram_rnw,
    ram_addr, ram_do, tdout);

5   input clk, nrst, resync, u_symbol, uc_pilot, upwstr, uprstr, te, tdin, upsel1,
    upsel2;
    input [1:0] guard;
    input [3:0] freq_sweep, sr_sweep, upaddr;
    input [11:0] ui_data, uq_data;
10  input [13:0] ram_do;
    output ram_rnw, tdout;
    output [9:0] ram_addr;
    output [12:0] freq_err, samp_err;
    output [13:0] ram_di;
15  inout [7:0] lupdata;

    wire got_phase, en, load, symbol, u_symbol_buf, uc_pilot_buf;
    wire freq_open, sample_open;
    wire [1:0] guard_reg;
20  wire [10:0] xcount;
    wire [11:0] ui_data_buf, uq_data_buf;
    wire [13:0] phase_in, phase_out;
    wire [20:0] acc_out_simple;
    wire [29:0] acc_out_prod;
25  wire [12:0] freq_err_uf, samp_err_uf;
    wire [12:0] freq_err_fil, samp_err_fil, freq_twiddle,
        sample_twiddle;

30  buffer buffer (.clk(clk), .nrst(nrst), .resync(resync), .u_symbol_in(u_symbol),
    .uc_pilot_in(uc_pilot), .ui_data_in(ui_data),
    .uq_data_in(uq_data), .u_symbol_out(u_symbol_buf),
    .uc_pilot_out(uc_pilot_buf), .ui_data_out(ui_data_buf),
    .uq_data_out(uq_data_buf), .got_phase(got_phase));
35  tan_taylor phase_extr (.clk(clk), .nrst(nrst), .resync(resync),
    .uc_pilot(uc_pilot_buf), .ui_data(ui_data_buf),
    .uq_data(uq_data_buf), .phase(phase_in),
    .got_phase(got_phase));
40  addr_gen addr_gen (.clk(clk), .resync(resync), .u_symbol(u_symbol_buf),
    .uc_pilot(uc_pilot_buf), .got_phase(got_phase), .en(en),
    .load(load), .guard(guard), .addr(ram_addr), .xcount(xcount),
    .guard_reg(guard_reg), .symbol(symbol));
45  pilot_store pilot_store (.clk(clk), .en(en), .ram_do(ram_do),
    .phase_in(phase_in), .ram_rnw(ram_rnw),
    .ram_di(ram_di), .phase_out(phase_out));

50  acc_simple acc_simple (.clk(clk), .resync(resync), .load(load),
    .symbol(symbol), .new_phase(phase_in),
    .old_phase(phase_out), .acc_out(acc_out_simple));

55  acc_prod acc_prod (.clk(clk), .resync(resync), .load(load),
    .symbol(symbol), .new_phase(phase_in),
    .old_phase(phase_out), .xcount(xcount),

```



```

        .acc_out(acc_out_prod));

slow_arith slow_arith (.acc_simple(acc_out_simple), .acc_prod(acc_out_prod),
    .guard(guard_reg), .freq_err_uf(freq_err_uf),
5      .samp_err_uf(samp_err_uf));

avg_8 # (14)
    lpf_freq (.clk(clk), .resync(resync), .symbol(symbol),
10      .in_data(freq_err_uf), .avg_out(freq_err_fil));

avg_8 # (14)
    lpf_samp (.clk(clk), .resync(resync), .symbol(symbol),
15      .in_data(samp_err_uf), .avg_out(samp_err_fil));

/* median_filter # (14)
    lpf_freq (.clk(clk), .nrst(nrst), .in_valid(symbol),
20      .din(freq_err_uf), .dout(freq_err_fil));

median_filter # (14)
    lpf_samp (.clk(clk), .nrst(nrst), .in_valid(symbol),
25      .din(samp_err_uf), .dout(samp_err_fil)); */

sweep_twiddle sweep_twiddle (.freq_err_fil(freq_err_fil),
30      .samp_err_fil(samp_err_fil),
    .freq_sweep(freq_sweep),
    .sr_sweep(sr_sweep), .freq_open(freq_open),
    .sample_open(sample_open),
    .freq_twiddle(freq_twiddle),
35      .sample_twiddle(sample_twiddle),
    .freq_err_out(freq_err),
    .samp_err_out(samp_err));

lupidec lupidec (.clk(clk), .nrst(nrst), .resync(resync), .upaddr(upaddr),
40      .upwstr(upwstr), .uprstr(uprstr), .lupdata(lupdata),
    .freq_open(freq_open), .sample_open(sample_open),
    .freq_twiddle(freq_twiddle), .sample_twiddle(sample_twiddle),
    .sample_loop_bw(), .freq_loop_bw(), .freq_err(freq_err),
    .samp_err(samp_err), .f_err_update(), .s_err_update());

45      endmodule

```

Listing 31

```

// SccsId: %W% %G%
45  /*****
    Copyright (c) 1997 Pioneer Digital Design Centre Limited
    *****/

50  module lupidec (clk, nrst, resync, upaddr, upwstr, uprstr, lupdata, freq_open,
    sample_open, freq_twiddle, sample_twiddle, sample_loop_bw,
    freq_loop_bw, freq_err, samp_err, f_err_update,
55      s_err_update);

    input clk, nrst, resync, upwstr, uprstr, f_err_update, s_err_update;

```

```

input [3:0] upaddr;
input [12:0] freq_err, samp_err;
inout [7:0] lupdata;
output freq_open, sample_open;
5 output [12:0] freq_twiddle, sample_twiddle, sample_loop_bw, freq_loop_bw;

reg freq_open, sample_open;
reg [12:0] freq_twiddle, sample_twiddle, sample_loop_bw, freq_loop_bw;

10 wire wr_str;
   wire [3:0] wr_addr;
   wire [7:0] wr_data;

15 /*FOLDBEGINS 0 2 "address decode"*/
   /*FOLDBEGINS 0 0 "read decode"*/
   wire f_err_h_ren = (upaddr == 4'he);
   wire f_err_l_ren = (upaddr == 4'hf);
   wire s_err_h_ren = (upaddr == 4'hc);
20 wire s_err_l_ren = (upaddr == 4'hd);
   wire f_twd_h_ren = (upaddr == 4'h4);
   wire f_twd_l_ren = (upaddr == 4'h5);
   wire s_twd_h_ren = (upaddr == 4'h8);
   wire s_twd_l_ren = (upaddr == 4'h9);
25 wire f_lbw_h_ren = (upaddr == 4'h6);
   wire f_lbw_l_ren = (upaddr == 4'h7);
   wire s_lbw_h_ren = (upaddr == 4'ha);
   wire s_lbw_l_ren = (upaddr == 4'hb);
   /*FOLDENDS*/

30 /*FOLDBEGINS 0 0 "write decode"*/
   wire f_twd_h_wen = (wr_addr == 4'h4);
   wire f_twd_l_wen = (wr_addr == 4'h5);
   wire s_twd_h_wen = (wr_addr == 4'h8);
35 wire s_twd_l_wen = (wr_addr == 4'h9);
   wire f_lbw_h_wen = (wr_addr == 4'h6);
   wire f_lbw_l_wen = (wr_addr == 4'h7);
   wire s_lbw_h_wen = (wr_addr == 4'ha);
   wire s_lbw_l_wen = (wr_addr == 4'hb);
40 /*FOLDENDS*/
   /*FOLDENDS*/

   /*FOLDBEGINS 0 2 "upi regs"*/
   /*FOLDBEGINS 0 0 "freq error status reg"*/
45 upi_status_reg2 fr_err (.clk(clk), .nrst(nrst), .status_value({3'b0, freq_err}),
      .capture_strobe(f_err_update), .read_strobe(uprstr),
      .reg_select_l(f_err_l_ren), .reg_select_h(f_err_h_ren),
      .lupdata(lupdata));
   /*FOLDENDS*/

50 /*FOLDBEGINS 0 0 "sample error status reg"*/
   upi_status_reg2 sr_err (.clk(clk), .nrst(nrst), .status_value({3'b0, samp_err}),
      .capture_strobe(s_err_update), .read_strobe(uprstr),
      .reg_select_l(s_err_l_ren), .reg_select_h(s_err_h_ren),
55 .lupdata(lupdata));
   /*FOLDENDS*/

```

```

/*FOLDBEGINS 0 0 "control regs write latch"*/
upi_write_latch #(3)
    write_lat (.clk(clk), .nrst(nrst), .lupdata(lupdata), .upaddr(upaddr),
5      .write_strobe(upwstr), .write_data(wr_data),
      .write_address(wr_addr), .write_sync(wr_str));
/*FOLDENDS*/

/*FOLDBEGINS 0 0 "freq twiddle etc rdbk regs"*/
10 upi_rdbk_reg freq_r_upper (.control_value({freq_open, 2'b0, freq_twiddle[12:8]}),
    .read_strobe(uprstr), .reg_select(f_twd_h_ren),
    .lupdata(lupdata));

upi_rdbk_reg freq_r_lower (.control_value(freq_twiddle[7:0]), .read_strobe(uprstr),
15    .reg_select(f_twd_l_ren), .lupdata(lupdata));
/*FOLDENDS*/

/*FOLDBEGINS 0 0 "samp twiddle etc rdbk regs"*/
upi_rdbk_reg samp_r_upper (.control_value({sample_open, 2'b0,
20 sample_twiddle[12:8]}),
    .read_strobe(uprstr), .reg_select(s_twd_h_ren),
    .lupdata(lupdata));

upi_rdbk_reg samp_r_lower (.control_value(sample_twiddle[7:0]),
25    .read_strobe(uprstr),
    .reg_select(s_twd_l_ren), .lupdata(lupdata));
/*FOLDENDS*/

/*FOLDBEGINS 0 0 "freq loop bw rdbk regs"*/
30 upi_rdbk_reg fr_lp_r_upper (.control_value({3'b0, freq_loop_bw[12:8]}),
    .read_strobe(uprstr), .reg_select(f_lbw_h_ren),
    .lupdata(lupdata));

upi_rdbk_reg fr_lp_r_lower (.control_value(freq_loop_bw[7:0]),
35    .read_strobe(uprstr), .reg_select(f_lbw_l_ren),
    .lupdata(lupdata));
/*FOLDENDS*/

/*FOLDBEGINS 0 0 "samp loop bw rdbk regs"*/
40 upi_rdbk_reg sr_lp_r_upper (.control_value({3'b0, sample_loop_bw[12:8]}),
    .read_strobe(uprstr), .reg_select(s_lbw_h_ren),
    .lupdata(lupdata));

upi_rdbk_reg sr_lp_r_lower (.control_value(sample_loop_bw[7:0]),
45    .read_strobe(uprstr), .reg_select(s_lbw_l_ren),
    .lupdata(lupdata));
/*FOLDENDS*/
/*FOLDENDS*/

50 /*FOLDBEGINS 0 2 "control regs"*/
always @ (posedge clk)
begin
    if (!nrst)
    begin
55      freq_open <= 0;
      sample_open <= 0;
    end
end

```

```

    freq_twiddle <= 0;
    sample_twiddle <= 0;
    sample_loop_bw <= 0; //????
    freq_loop_bw <= 0; //????
5   end
    else
    begin
        if (wr_str)
        begin
10         if (f_twd_h_wen)
            begin
                freq_open <= wr_data[7];
                freq_twiddle[12:8] <= wr_data[4:0];
            end
15         if (f_twd_l_wen)
            freq_twiddle[7:0] <= wr_data[7:0];

            if (s_twd_h_wen)
20         begin
                sample_open <= wr_data[7];
                sample_twiddle[12:8] <= wr_data[4:0];
            end

            if (s_twd_l_wen)
25         sample_twiddle[7:0] <= wr_data[7:0];

            if (f_lbw_h_wen)
30         freq_loop_bw[12:8] <= wr_data[4:0];

            if (f_lbw_l_wen)
                freq_loop_bw[7:0] <= wr_data[7:0];

            if (s_lbw_h_wen)
35         sample_loop_bw[12:8] <= wr_data[4:0];

            if (s_lbw_l_wen)
                sample_loop_bw[7:0] <= wr_data[7:0];

40         end
        end
    end
    /*FOLDENDS*/
45 endmodule

```

Listing 32

```

50 // SccsId: %W% %G%
    /*****
    Copyright (c) 1997 Pioneer Digital Design Centre Limited
    *****/
55

```

```

module pilot_store (clk, en, ram_do, phase_in, ram_rnw, ram_di, phase_out);

    input clk, en;
    // input [9:0] addr;
5    input [13:0] phase_in;
    input [13:0] ram_do;
    output ram_rnw;
    output [13:0] ram_di, phase_out;

10   wire ram_rnw;
    // reg en_d1;
    // reg [9:0] addr_reg;
    // reg [13:0] mem [579:0];
    reg [13:0] phase_out; //, phase_in_reg;
15   wire [13:0] ram_di;

    always @ (posedge clk)
    begin
20     // en_d1 <= en;

        if (en)
        begin
25         // phase_in_reg <= phase_in;
            // addr_reg <= addr;
            phase_out <= ram_do;
            // phase_out <= mem[addr];
            end
            // if (en_d1)
30         // mem[addr_reg] <= phase_in_reg;
            end

        assign ram_di = phase_in;
        assign ram_rnw = !en;
35   endmodule

```

Listing 33

```

40 // ScclId: %W% %G%
//*****
    Copyright (c) 1997 Pioneer Digital Design Centre Limited
    *****/

45

module slow_arith (acc_simple, acc_prod, guard, freq_err_uf, samp_err_uf);

50   input [1:0] guard;
    input [20:0] acc_simple;
    input [29:0] acc_prod;
    output [12:0] freq_err_uf, samp_err_uf;

55   reg [12:0] freq_err_uf, samp_err_uf;
    reg [20:0] freq_scale;
    reg [38:0] inter_freq;

```

```

reg sign;
reg [20:0] mod_acc;
reg [38:0] mod_trunc_sat;
reg [41:0] mod;
5
reg sign_a, sign_b, sign_inter_sr;
reg [20:0] mod_acc_s;
reg [29:0] mod_acc_p;
reg [35:0] a, mod_a;
10 reg [35:0] b, mod_b;
reg [36:0] mod_diff, diff;
reg [46:0] inter_sr, mod_inter_sr;

15 parameter sp = 45, acc_x = 33927, samp_scale = 11'b10100100110;

always @ (guard)
case (guard)
20 2'b00: freq_scale = 21'b011110100111110001011; // guard == 64
2'b01: freq_scale = 21'b0111101101110001000011; // guard == 128
2'b10: freq_scale = 21'b011100000100011101010; // guard == 256
2'b11: freq_scale = 21'b011001010000110011111; // guard == 512
endcase

25 always @ (acc_simple or freq_scale)
begin

sign = acc_simple[20];
mod_acc = sign ? (~acc_simple + 1) : acc_simple;
30 mod = (freq_scale * mod_acc);
// inter_freq = sign ? (~mod + 1) : mod;

if (mod[41:38] > 0)
begin
35 mod_trunc_sat = 39'h3fffffff;
$display("freq_err saturated");
end
else
mod_trunc_sat = mod[38:0];
40
inter_freq = sign ? (~mod_trunc_sat + 1) : mod_trunc_sat;

freq_err_uf = inter_freq >> 26;
end

45
always @ (acc_simple or acc_prod)
begin

sign_a = acc_prod[29];
50 mod_acc_p = sign_a ? (~acc_prod + 1) : acc_prod;
mod_a = sp * mod_acc_p;
a = sign_a ? (~mod_a + 1) : mod_a;

sign_b = acc_simple[20];
55 mod_acc_s = sign_b ? (~acc_simple + 1) : acc_simple;
mod_b = acc_x * mod_acc_s;

```

```

b = sign_b ? (~mod_b + 1) : mod_b;
diff = {a[35], a} - {b[35], b}; // sign extend
5  sign_inter_sr = diff[36];
   mod_diff = sign_inter_sr ? (~diff + 1) : diff;
   mod_inter_sr = (mod_diff * samp_scale);
   inter_sr = sign_inter_sr ? (~mod_inter_sr + 1) : mod_inter_sr;
10  samp_err_uf = inter_sr >> 34; ///scaling!!
   end

endmodule

15  // SccsId: %W% %G%
   /*****
   Copyright (c) 1997 Pioneer Digital Design Centre Limited
   *****/
20  module sweep_twiddle (freq_err_fil, samp_err_fil, freq_sweep, sr_sweep,
   freq_open, sample_open, freq_twiddle, sample_twiddle,
   freq_err_out, samp_err_out);

25  input freq_open, sample_open;
   input [3:0] freq_sweep, sr_sweep;
   input [12:0] freq_err_fil, samp_err_fil, freq_twiddle, sample_twiddle;
   output [12:0] freq_err_out, samp_err_out;

30  reg [12:0] freq_err_out, samp_err_out;
   reg [12:0] freq_err_swept, samp_err_swept;

   always @ (freq_sweep or freq_err_fil)
35  case (freq_sweep)
    4'b0000: freq_err_swept = freq_err_fil;
    4'b0001: freq_err_swept = freq_err_fil + 500;
    4'b0010: freq_err_swept = freq_err_fil + 1000;
    4'b0011: freq_err_swept = freq_err_fil + 1500;
40  4'b0100: freq_err_swept = freq_err_fil + 2000;
    4'b0101: freq_err_swept = freq_err_fil + 2500;
    4'b0110: freq_err_swept = freq_err_fil + 3000;
    4'b0111: freq_err_swept = freq_err_fil + 3500;
    default: freq_err_swept = freq_err_fil;
45  endcase

   always @ (sr_sweep or samp_err_fil)
   case (sr_sweep)
50  4'b0000: samp_err_swept = samp_err_fil;
    4'b0001: samp_err_swept = samp_err_fil + 500;
    4'b0010: samp_err_swept = samp_err_fil - 500;
    4'b0011: samp_err_swept = samp_err_fil + 1000;
    4'b0100: samp_err_swept = samp_err_fil - 1000;
    4'b0101: samp_err_swept = samp_err_fil + 1500;
55  4'b0110: samp_err_swept = samp_err_fil - 1500;
    4'b0111: samp_err_swept = samp_err_fil + 2000;

```

Listing 34

```

4'b1000: samp_err_swept = samp_err_fil - 2000;
default: samp_err_swept = samp_err_fil;
endcase

```

```

5  always @ (freq_err_swept or freq_open or freq_twiddle)
    if (freq_open)
        freq_err_out = freq_twiddle;
    else
10   freq_err_out = freq_err_swept + freq_twiddle;

    always @ (samp_err_swept or sample_open or sample_twiddle)
    if (sample_open)
        samp_err_out = sample_twiddle;
    else
15   samp_err_out = samp_err_swept + sample_twiddle;

```

```

endmodule

```

Listing 35

```

20  // SccsId: %W% %G%
    /*****
    Copyright (c) 1997 Pioneer Digital Design Centre Limited
25  *****/

module tan_taylor (clk, nrst, resync, uc_pilot, ui_data, uq_data, phase,
30   got_phase);

    input clk, nrst, resync, uc_pilot;
    input [11:0] ui_data, uq_data;
    output got_phase;
35   output [13:0] phase;

    reg got_phase;
    reg [13:0] phase;
    reg add, qgti, modqeqi, i_zero_reg, q_zero_reg, go;
40   reg [1:0] quadrant;
    reg [6:0] count, count_d1;
    reg [10:0] mod_i, mod_q, coeff, numer, denom;
    reg [21:0] x_sqd, x_pow, next_term, sum, flip, next_term_unshift, prev_sum,
        x_sqd_unshift, x_pow_unshift;
45   wire got;
    wire [10:0] div;

    parameter pi = 6434, pi_over2 = 3217, minus_pi_o2 = 13167, pi_over4 = 1609;

50   divide div1 (clk, go, numer, denom, div, got);

    always @ (posedge clk)
    begin
55   if (!nrst || resync)
        count <= 7'b1111111;
    end

```



```

else
begin
  if (uc_pilot)
  begin
5    mod_i <= ui_data[11] ? (~ui_data[10:0] + 1) : ui_data[10:0];
    mod_q <= uq_data[11] ? (~uq_data[10:0] + 1) : uq_data[10:0];
    quadrant <= {uq_data[11], ui_data[11]};
    count <= 0;
    go <= 0;
10   end

  else
  begin
    if (count == 0)
15    begin
      qgti <= (mod_q > mod_i);
      modqeqi <= (mod_q == mod_i);
      i_zero_reg <= (mod_i == 0);
      q_zero_reg <= (mod_q == 0);
20    add <= 0;
      go <= 1;
      count <= 1;
    end

25    if ((count >= 3) && (count < 71))
      count <= count + 2;

    if (count == 1)
    begin
30      go <= 0;
      if (got)
      begin
        sum <= div;
        x_pow <= div;
35      x_sqd <= x_sqd_unshift >> 11;
        count <= 3;
      end
    end

40    if ((count > 1) && (count < 69))
      x_pow <= x_pow_unshift >> 11;
    if ((count > 3) && (count < 69))
      next_term <= next_term_unshift >> 12;
    if ((count > 5) && (count < 69))
45    begin
      prev_sum <= sum;
      sum <= add ? (sum + next_term) : (sum - next_term);
      add <= !add;
    end
50    end
    if (count == 67)
      sum <= (prev_sum + sum) >> 1;
    if (count == 69)
    casex ({i_zero_reg, q_zero_reg, qgti, modqeqi, quadrant})
55    6'b1xx0_0x: phase <= pi_over2;
    6'b1xx0_1x: phase <= minus_pi_o2;

```

```

6'b01x0_x0: phase <= 0;
6'b01x0_x1: phase <= pi;

5   6'b0010_00: phase <= {2'b00, flip[11:0]};
    6'b0010_01: phase <= pi - {2'b00, flip[11:0]};
    6'b0010_10: phase <= 0 - {2'b00, flip[11:0]};
    6'b0010_11: phase <= {2'b00, flip[11:0]} - pi;

10  6'b0000_00: phase <= {2'b00, sum[11:0]};
    6'b0000_01: phase <= pi - {2'b00, sum[11:0]};
    6'b0000_10: phase <= 0 - {2'b00, sum[11:0]};
    6'b0000_11: phase <= {2'b00, sum[11:0]} - pi;

15  6'bxxx1_00: phase <= pi_over4;
    6'bxxx1_01: phase <= pi - pi_over4;
    6'bxxx1_10: phase <= 0 - pi_over4;
    6'bxxx1_11: phase <= pi_over4 - pi;
    endcase

20  count_d1 <= count;
    got_phase <= (count == 69);
    end
end

25  always @(div)
    x_sqd_unshift = div * div; // had to do this in order to stop synthesis throwing away!

    always @(x_pow or coeff)
    next_term_unshift = (x_pow * coeff); // compass dp_cell mult_booth_csum

30  always @(x_pow or x_sqd)
    x_pow_unshift = (x_pow * x_sqd); // compass dp_cell mult_booth_csum

    always @(count_d1)
    case (count_d1)
35      3: coeff = 11'b10101010101;
        5: coeff = 11'b01100110011;
        7: coeff = 11'b01001001001;
        9: coeff = 11'b00111000111;
40      11: coeff = 11'b00101110100;
        13: coeff = 11'b00100111011;
        15: coeff = 11'b00100010001;
        17: coeff = 11'b00011110001;
        19: coeff = 11'b00011010111;
45      21: coeff = 11'b00011000011;
        23: coeff = 11'b00010110010;
        25: coeff = 11'b00010100011;
        27: coeff = 11'b00010010111;
        29: coeff = 11'b00010001101;
50      31: coeff = 11'b00010000100;
        33: coeff = 11'b00001111100;
        35: coeff = 11'b00001110101;
        37: coeff = 11'b00001101110;
        39: coeff = 11'b00001101001;
55      41: coeff = 11'b00001100100;
        43: coeff = 11'b00001011111;

```

```

45: coeff = 11'b00001011011;
47: coeff = 11'b00001010111;
49: coeff = 11'b00001010011;
51: coeff = 11'b00001010000;
53: coeff = 11'b00001001101;
55: coeff = 11'b00001001010;
57: coeff = 11'b00001000111;
59: coeff = 11'b00001000101;
61: coeff = 11'b00001000011;
63: coeff = 11'b00001000001;
// 65: coeff = 11'b00000111111;
// 67: coeff = 11'b00000111101;
// 69: coeff = 11'b00000111011;
// 71: coeff = 11'b00000111001;
// 73: coeff = 11'b00000111000;
// 75: coeff = 11'b00000110110;
// 77: coeff = 11'b00000110101;
default: coeff = 11'bx;
endcase

always @ (mod_q or mod_i or qgti)
begin
    numer = qgti ? mod_i : mod_q;
    denom = qgti ? mod_q : mod_i;
end

always @ (sum)
    flip = pi_over2 - sum;

// always @ (got)
// if (got)
// $display("numer was %d, denom was %d, div then %d", numer, denom, div);

// always @ (count)
// if (count < 68 ) $display("as far as x to the %0d term, approx = %d", (count-6),
sum);

always @ (got_phase)
begin: display
    reg [13:0] real_phase;

    if (phase[13])
    begin
        real_phase = (~phase + 1);
        if (got_phase) $display("%t: got phase, phase = -%0d", $time, real_phase);
    end
    else
    begin
        if (got_phase) $display("%t: got phase, phase = %0d", $time, phase);
    end
end // display

endmodule

```

While this invention has been explained with reference to the structure disclosed herein, it is not confined to the details set forth and this application is intended to cover any modifications and changes as may come within the scope of the following claims:

CLAIMS

1 1. A digital receiver for multicarrier signals comprising:
2 an amplifier accepting an analog multicarrier signal, wherein said multicarrier
3 signal comprises a stream of data symbols having a symbol period T_s , wherein the
4 symbols comprise an active interval, a guard interval, and a boundary therebetween,
5 said guard interval being a replication of a portion of said active interval;
6 an analog to digital converter coupled to said amplifier;
7 an I/Q demodulator for recovering in phase and quadrature components from
8 data sampled by said analog to digital converter;
9 an automatic gain control circuit coupled to said analog to digital converter for
10 providing a gain control signal for said amplifier;
11 a low pass filter circuit accepting I and Q data from said I/Q demodulator, wherein
12 said I and Q data are decimated;
13 a resampling circuit receiving said decimated I and Q data at a first rate and
14 outputting resampled I and Q data at a second rate;
15 an FFT window synchronization circuit coupled to said resampling circuit for
16 locating a boundary of said guard interval;
17 a real-time pipelined FFT processor operationally associated with said FFT
18 window synchronization circuit, wherein said FFT processor comprises at least one
19 stage, said stage comprising:
20 a complex coefficient multiplier; and
21 a memory having a lookup table defined therein for multiplicands being
22 multiplied in said complex coefficient multiplier, a value of each said multiplicand
23 being unique in said lookup table; and
24 a monitor circuit responsive to said FFT window synchronization circuit for
25 detecting a predetermined event, whereby said event indicates that a boundary between
26 an active symbol and a guard interval has been located.

1 2. The receiver according to claim 1, wherein said FFT window synchronization
2 circuit comprises:
3 a first delay element accepting currently arriving resampled I and Q data, and
4 outputting delayed resampled I and Q data;
5 a subtracter, for producing a difference signal representative of a difference
6 between said currently arriving resampled I and Q data and said delayed resampled I
7 and Q data;

8 a first circuit for producing an output signal having a unipolar magnitude that is
9 representative of said difference signal of said subtracter;
10 a second delay element for storing said output signal of said first circuit;
11 a third delay element receiving delayed output of said second delay element; and
12 a second circuit for calculating a statistical relationship between data stored in
13 said second delay element and data stored in said third delay element and having an
14 output representative of said statistical relationship.

1 3. The receiver according to claim 2, wherein said statistical relationship
2 comprises an F ratio.

1 4. The receiver according to claim 1, wherein said FFT processor operates in an
2 8K mode.

1 5. The receiver according to claim 1, wherein said wherein said FFT processor
2 further comprises an address generator for said memory, said address generator
3 accepting a signal representing an order dependency of a currently required multipli-
4 cand, and outputting an address of said memory wherein said currently required
5 multiplicand is stored.

1 6. The receiver according to claim 5, wherein each said multiplicand is stored in
2 said lookup table in order of its respective order dependency for multiplication by said
3 complex coefficient multiplier, said order dependencies of said multiplicands defining an
4 incrementation sequence, and said address generator comprises:

5 an accumulator for storing a previous address that was generated by said
6 address generator;

7 a circuit for calculating an incrementation value of said currently required
8 multiplicand; and

9 an adder for adding said incrementation value to said previous address.

1 7. The receiver according to claim 6, wherein said lookup table comprises a
2 plurality of rows, and said incrementation sequence comprises a plurality of
3 incrementation sequences, said multiplicands being stored in row order, wherein

4 in a first row a first incrementation sequence is 0;

5 in a second row a second incrementation sequence is 1;

6 in a third row first and second break points B1, B2 of a third incrementation
7 sequence are respectively determined by the relationships